COMMODORE 64
GUIDA PER L'USO

Pubblicato dalla Commodore Italiana S.r.l.

AVVERTENZE

Questa apparecchiatura genera ed usa energia a radiofrequenza e se non è installata ed usata correttamente e cioè in stretta conformità con le istruzioni del fabbricante, può provocare interferenze alla ricezione radio e televisiva. Essa è stata omologata ed è risultata conforme ai limiti per il dispositivo di calcolo di Classe B e conforme alle specifiche nella Subpart J della Part 15 delle regole FCC che sono intese a fornire ragionevole protezione a fronte di tale interferenza nelle installazioni residenziali. In ogni caso non c'è garanzia che non si verifichi interferenze in una particolare installazione. Se questa apparecchiatura provoca interferenza alla ricezione radio o televisiva, che può essere determinata spegnendo e riaccendendo l'apparecchiatura stessa, l'utente è invitato a cercare di correggere l'interferenza mediante una o più delle seguenti misure:

- riorientare l'antenna ricevente
- riposizionare il computer rispetto al ricevitore
- spostare il computer allontanandolo dal ricevitore
- collegare il computer a prese diverse in modo che computer e ricevitore siano alimentati da circuiti derivati diversi.

Se necessario l'utente dovrà consultare il rivenditore o un tecnico radiotelevisivo esperto per ulteriori suggerimenti. L'utente potrà trovare utile il seguente opuscolo preparato dalla Federal Communications Commission: "How to Identify and Resolve Radio-TV Interference Problems", che può essere richiesto all'U.S. Government Printing Office, Washington, D.C. 20402, N. codice 004-000-00345-4.

TAVOLA DEI CONTENUTI

INTRODUZIONE

Siete ora l'orgoglioso proprietario dei COMMODORE 64, per cui vi facciamo le nostre più vive congratulazioni per aver acquistato uno dei migliori computer del mondo. La COMMODORE è nota come la società del computer amico ed essere amici significa fornire manuali di istruzione facili da leggere, da comprendere e da usare. La GUIDA PER L'USO DEL COMMODORE 64 contiene tutte le informazioni necessarie per disporre opportunamente l'apparecchiatura, prendere conoscenza con il COMMODORE 64 ed avviarvi in maniera facile e divertente ad imparare a compilare i propri programmi.

Per coloro che non intendono imparare la programmazione, abbiamo inserito tutte le informazioni che vi accorrono per usare i programmi COMMODORE o altri programmi pronti e/o cassette per giochi (software di terzi) nella parte iniziale. Ciò significa che non occorre effettuare ricerche in tutto il manuale per cominciare ad usare il computer.

Diamo ora uno sguardo ad alcune delle interessanti caratteristiche che vi stanno aspettando all'interno del vostro COMMODORE 64. Innanzitutto se dovete costruire dei grafici, avete a disposizione il "costruttore di immagini", più avanzato di tutta l'industria dei microcomputer, che abbiamo chiamato GRAFICI DI ANIMAZIONE e che vi consente di disegnare vostre immagini in quattro colori diversi, esattamente come quelle che vedete nei videogiochi delle sale specializzate. Non solo, ma l'EDITOR DI ANIMAZIONE vi consente di animare fino a 8 diversi livelli di immagini alla volta. Potete cioè spostare le vostre creazioni ovunque sullo schermo, addirittura far passare un'immagine davanti o dietro ad un'altra. Il vostro COMMODORE 64 consente inoltre il rilevamento automatico di collisione che istruisce il computer ad intraprendere l'azione appropriata quando le immagini animate si urtano l'una con l'altra.

Il COMMODORE 64 dispone inoltre di effetti musicali sonori incorporati le cui capacità sfidano quelle di molti sintetizzatori musicali. Potete così disporre di tre voci o timbri indipendenti, ciascuna con un campo di 9 ottave intere "tipo pianoforte". Inoltre avete la possibilità di lavorare con quattro diverse forme d'onda (a dente di sega, a triangolo, ad impulso variabile e di rumore), un generatore ADSR programmabile (salita, discesa, piano, smorzo), un generatore di inviluppi, filtri programmabili alti, bassi e passabanda per ciascuna voce nonché regolazioni di volume e di risonanza variabili. E se volete che la vostra musica venga riprodotta con attrezzature sonore professionali, COMMODORE 64 vi consente di collegare la vostra uscita audio a pressoché qualsiasi sistema di amplificazione di alta qualità.

E giacché stiamo parlando di collegare il COMMODORE 64 ad altre apparecchiature... cogliamo l'occasione per dire che il vostro sistema può essere ampliato aggiungendogli accessori, noti come periferiche, man mano che le vostre esigenze di calcolo crescono. Alcune delle opzioni previste comprendono un registratore DATASSETTE* o fino a cinque unità di memoria a dischi VIC 1541 per i programmi da voi realizzati e/o che volete eseguire. Se disponete già di un'unità disco VIC 1540 il rivenditore potrà aggiornarla per l'impiego con COMMODORE 64.

Potete aggiungere una stampante a matrice VIC per ottenere copie stampate o tabulati di programmi, lettere, fatture, ecc... Se volete collegarvi con computer più potenti e con le relative massicce data base vi basta inserire ad innesto un caricatore VICMODEM per avere a disposizione i servizi di centinaia di specialisti e per accedere a numerose reti di informazioni utilizzando il telefono d'ufficio o di casa. Infine, se siete interessanti agli infiniti

programmi applicativi disponibili in CP/M**, potete munire COMMODORE 64 di un microprocessore a innesto Z-80.

Altrettanto importante di tutto l'hardware disponibile è il fatto che questa GUIDA PER L'USO sia in grado di aiutarvi a sviluppare la vostra conoscenza del computer. Essa non vi dirà tutto ciò che dovete sapere sui computer ma vi rinvierà a numerose pubblicazioni dalle quali potrete attingere informazioni più dettagliate sugli argomenti di vostro interesse. Noi vogliamo che vi godiate veramente il vostro nuovo COMMODORE 64. E per divertirvi, ricordate: la programmazione non è i! tipo di cosa che potete imparare in un giorno. Siate pazienti e leggete a fondo la GUIDA PER L'USO. Ma prima di iniziare, dedicate qualche minuto a compilare ed a spedire la cartolina di registrazione fornita insieme al computer. Ciò vi assicura che il vostro COMMODORE 64 sia correttamente registrato presso la sede della COMMODORE in modo da poter ricevere le informazioni più aggiornate riguardanti i miglioramenti futuri che lo riguardano. Buon divertimento!

NOTA: Molti programmi sono ancora in fase di sviluppo nel momento in cui questo manuale viene stampato. Vi consigliamo quindi di rimanere in contatto con il rivenditore locale COMMODORE e con i vari club e riviste di utenti Commodore che vi terranno aggiornati sulla enorme quantità di programmi applicativi che vengono scritti in tutto il mondo per COMMODORE 64.

* DATASSETTE è un marchio di fabbrica registrato della COMMODORE Business Machines, Inc.
** CP/M è un marchio di fabbrica registrato della Digital Research Inc. Le specifiche possono essere variate senza preavviso.

CAPITOLO 1

PRIMI APPROCCI CON IL COMMODORE 64

CONTENUTO DELL'IMBALLO E COLLEGAMENTO DEL COMMODORE 64

Le seguenti istruzioni mostrano come collegare il COMMODORE 64 ad un comune televisore, ad un sistema sonoro o ad un monitor. Prima di collegare qualsiasi cosa al computer controllare il contenuto dell'imballo del COMMODORE 64: insieme a questo manuale, si dovrebbe trovare quanto segue:

1. COMMODORE 64
2. Alimentatore (piccola scatola con una spina di alimentazione AC e un cavo con la presa per il COMMODORE 64)
3. Cavo coassiale per antenna

Se uno qualsiasi di questi componenti è mancante rivolgiti immediatamente al tuo rivenditore.
Vediamo innanzitutto dove sono sistemate e come funzionano le varie porte di questo computer.

CONNETTORI SITUATI SULLA PARTE DESTRA DEL COMMODORE 64

1. **Presa di alimentazione**, la presa del cavo di alimentazione va collegata qui.
2. **Interruttore di alimentazione.**
3. **Porta per il comando giochi**, ad ognuno di questi connettori può essere collegato un joystick, paddle o una penna ottica.

CONNETTORI POSTERIORI

4. **Connettore per cartuccia**, questa porta rettangolare accetta programmi o giochi memorizzati su cartucce.

5. **Connettore-TV**, questo connettore fornisce sia il segnale video che quello audio all'input antenna (75 ohm) del televisore.

6. **Audio e video output**, questa porta fornisce direttamente il segnale audio che può essere collegato ad un sistema Hi-Fi. Mette anche a disposizione un segnale video che può pilotare un TV-monitor.

[IMMAGINE PAG 3]

7. **Porta seriale**, alla quale si può collegare una stampante e un floppy disk singolo.

8. **Interfaccia per registratore**, un registratore a cassette può essere collegato al computer per poter salvare e poi caricare programmi e dati in memoria.

9. **User port**, porta libera e programmabile di input/output e allo stesso tempo connettore per cartucce ad inserimento come l'interfaccia RS232.

INSTALLAZIONE

Collegamento ad un comune televisore

1. Uso dell'UHF input di antenna.

Collegare il cavo antenna fornito, al connettore TV posto sulla parte posteriore del COMMODORE 64, e la parte opposta del cavo alla presa del televisore. Sintonizzarsi sul canale 36.

[IMMAGINE PAG 4]

2. Uscita video

Usando un TV-monitor, il miglior risultato si ottiene se questo è a colori. Occorre un cavo coassiale (non fornito) con una spina a 5 poli (DIN 41524) che va inserita nel connettore 6 del COMMODORE 64. Sul lato opposto del cavo, occorre un connettore video (DIN 45322) da inserire nel monitor. Se il televisore e già predisposto con una presa video, la si può usare anche come monitor. Per fare questo bisogna assicurarsi che il connettore video del televisore sia usato come «input».

Per commutare questo connettore come input, viene fornito un voltaggio ausiliario di 12 V, al polo 1 del medesimo; ma il COMMODORE 64 non fornisce questo voltaggio. Si prega di consultare un tecnico-TV specializzato, per le modifiche necessarie. Alcuni televisori richiedono soltanto un ponticello all'interno della presa video, in quanto forniscono il voltaggio ausiliario richiesto, sul polo 5 del connettore.

ATTENZIONE, in nessuna circostanza questi 12 V devono arrivare al COMMODORE 64, se ciò accadesse il computer sarà danneggiato immediatamente. Quindi è preferibile rivolgersi ad un tecnico specializzato.

Per ricevere i normali programmi televisivi bisogna disconnettere questo cavo del televisore

[IMMAGINE PAG 5]

3. Connessione alimentazione

Collegare l'alimentatore al COMMODORE 64 (connettore 1) ed inserire la spina-AC alla presa a muro.

4. Collegamenti opzionali

Il COMMODORE 64 fornisce un canale audio in alta fedeltà, quindi si ha la possibilità, se desiderato, di collegarlo con un amplificatore di qualità. Il segnale sonoro si trova sul polo 3 del connettore 6 (vedi figura in basso).

[IMMAGINE PAG 6]

Il COMMODORE 64 ha la possibilità di usare come unità periferiche il floppy disk singolo VIC 1541 e la stampante VIC 1525.

Questo sistema completo viene mostrato nella figura in basso.

[IMMAGINE PAG 7]

COLLAUDO

1. Accendere il computer usando l'interruttore situato sulla parte destra della tastiera.
2. Dopo alcuni secondi il seguente messaggio comparirà sullo schermo.

```
    **** COMMODORE 64 BASIC V2 ****
64K RAM SYSTEM   38911 BASIC BYTES FREE

READY

|
+--- Il cursore lampeggiante sta a significare che il COMMODORE 64 è pronto per la programmazione
```

3. Se il televisore ha il pomello per la sintonia manuale, muoverlo fino a quando si ottiene una immagine limpida. Se invece il televisore ha un AFC, si sintonizzerà automaticamente.
4. Si può anche aggiustare il contrasto ed il colore così da ottenere la migliore visualizzazione sul televisore.

Se non si ottengono i risultati previsti controllare ancora una volta i collegamenti e i cavi. La tavola seguente sarà di aiuto per isolare qualsiasi problema.

Sintomo	Causa	Rimedi
La spia di accensione è spenta	Computer spento	Assicurarsi che l'interruttore sia nella posizione ON
	Cavo di alimentazione non collegato	Controllare se la presa di alimentazione ha falsi contatti
	Il fusibile è saltato	Rivolgersi ad un centro autorizzato per la sostituzione del fusibile
L'immagine video non compare sullo schermo	TV sintonizzato sul canale sbagliato	Controllare su altri canali per ottenere l'immagine
	Collegamento incorretto	Il computer si collega al terminale dell'antenna UHF
	Schermo non collegato	Controllare il collegamento dell'output video
A cartuccia inserita il segnale video è disturbato	Cartuccia inserita incorrettamente	Reinserire la cartuccia dopo aver spento il computer
Schermo in bianco/nero o con colori sbiaditi	TV non sintonizzato perfettamente	Sintonizzare la TV

Immagine con eccessivi rumori di fondo	Il volume della TV è troppo alto	Abbassare il volume della TV
Immagine perfetta, ma assenza del sonoro	Il volume della TV è troppo basso	Alzare il volume della TV
	Output ausiliario collegato impropriamente	Collegare il jack audio dell'input ausiliario selezionato dell'amplificatore

REGOLAZIONE DEI COLORI

C'è un modo semplice per ottenere un profilo di colori sul televisore in modo da poter facilmente regolare l'apparecchio. Quantunque per il momento non si abbia ancora familiarità con il funzionamento del computer, basta procedere per vedere come è facile usare il COMMODORE 64.

Osservare innanzitutto il lato sinistro della tastiera ed individuare il tasto contrassegnato **[CTRL].** La scritta è l'abbreviazione di ConTRoL e questo tasto viene usato unitamente ad altri per istruire il computer a svolgere un compito specifico.

[IMMAGINE PAG 10]

Per usare una funzione di controllo occorre tenere abbassato il tasto **[CTRL]** ed abbassare contemporaneamente un secondo tasto.

Provare in questo modo: tenere abbassato il tasto **[CTRL]** mentre si preme contemporaneamente il tasto **[9]**. Sollevare quindi entrambi i tasti. Non dovrebbe essere successo nulla di ovvio ma se ora si preme qualsiasi altro tasto, lo schermo mostra il carattere visualizzato in negativo anziché nel carattere normale - come il messaggio di apertura o qualsiasi cosa che è stata battuta precedentemente.

Tenere abbassato il tasto **[SPACE BAR].** Cosa succede? Se è stata eseguita la suddetta procedura correttamente, fintantoché rimane abbassata la barra di spazio **[SPACE BAR]** si dovrebbe vedere una barra azzurra spostarsi attraverso lo schermo e quindi abbassarsi alla riga successiva.

```
   **** COMMODORE 64 BASIC V2 ****
64K RAM SYSTEM  38911 BASIC BYTES FREE

READY
____________________________
__________
```

Ora, tenere abbassato **[CTRL]** mentre si preme uno qualsiasi degli altri tasti numerici, ciascuno dei quali è contrassegnato nella parte anteriore da un colore. Qualsiasi cosa verrà visualizzata da questo punto in avanti sarà in quel colore. Per esempio, tenere abbassato **[CTRL]** e il tasto **[9]** e quindi rilasciarli entrambi. Premere ora il tasto **[SPACE BAR].**

Osservare lo schermo. La barra è di color giallo! In maniera analoga è possibile cambiare il colore della barra scegliendolo tra quelli indicati sui tasti numerici premendo **[CTRL]** il tasto appropriato.

Provare a cambiare per due o tre volte il colore della barra, quindi regolare luminosità e contrasto del televisore in modo che ciò che si vede sullo schermo corrisponda ai colori scelti.

Lo schermo dovrebbe apparire più o meno come segue:

```
   **** COMMODORE 64 BASIC V2 ****
64K RAM SYSTEM  38911 BASIC BYTES FREE

_________________________     <------ [3] Barra Rossa
```

```
<------ [3],[6],[7] Barre Rossa, Verde, Blu
<------ [7],[8] Barre Blu, Gialla
<------ [8] Barra Gialla
```

A questo punto tutto è correttamente regolato e funziona perfettamente. I capitoli seguenti presenteranno il linguaggio BASIC. In ogni caso è possibile iniziare immediatamente ad usare alcune delle applicazioni «pronte» e i giochi disponibili per il COMMODORE 64 senza conoscere nulla sulla programmazione di computer.

Ciascuno di questi «packages» contiene informazioni dettagliate sul modo in cui usare il programma. Si suggerisce comunque di legger prima i capitoli di questo manuale per prendere maggiore familiarità con il funzionamento base del nuovo sistema.

CAPITOLO 2

PER COMINCIARE

- La tastiera
- Ritorno al funzionamento normale
- Caricamento e salvataggio di programmi
- PRINT e calcoli
- Precedenza
- Come combinare le cose

LA TASTIERA

Ora che tutto è messo a punto e regolato, occorre dedicare qualche instante e prendere familiarità con la tastiera che è il più importante mezzo di comunicazione con il COMMODORE 64.
Sotto molti aspetti la tastiera è simile di una normale macchina da scrivere. Ci sono comunque numerosi nuovi tasti che controllano funzioni specializzate. Segue ora una breve descrizione dei vari tasti e delle rispettive funzioni. Il funzionamento dettagliato di ciascun tasto sarà illustrato in successivi capitoli.

[IMMAGINE PAG 14]

[RETURN]
Il tasto **[RETURN]** segnala al computer di osservare l'informazione battuta ed immette quell'informazione in memoria.

[SHIFT]
Il tasto **[SHIFT]** funziona come quello di una normale macchina da scrivere. Molti tasti sono in grado di visualizzare due lettere o simboli e due caratteri grafici. Nel modo «maiuscolo/minuscolo» il tasto **[SHIFT]** fornisce i caratteri standard maiuscoli. Nel modo «maiuscolo/grafici» il tasto visualizza il carattere grafico riportato sul lato destro del tasto.
Nel caso dei tasti speciali di funzione, il tasto **[SHIFT]** dà la funzione contrassegnata sulla parte superiore del tasto stesso.

LE CORREZIONI

Nessuno è perfetto e COMMODORE 64 lo sa. Un certo numero di tasti di correzione consente di rimediare

agli errori di battitura e di spostare le informazioni sullo schermo.

[CRSR]

Ci sono dei tasti contrassegnati **[CRSR]** (CuRSoR-cursore), uno con freccia verso l'alto e verso il basso **[CRSR UP/DOWN]** l'altro con frecce verso destra e verso sinistra **[CRSR LEFT/RIGHT].** È possibile usare questi tasti per spostare il cursore verso l'alto e verso il basso o verso sinistra e verso destra. Nel modo non preceduto da **[SHIFT],** i tasti **[CRSR]** consentono di spostare il cursore verso il basso e verso destra. L'uso contemporaneo dei tasti **[SHIFT]** e **[CRSR]** consente di spostare il cursore verso l'alto o verso sinistra. I tasti di spostamento del cursore dispongono di una speciale funzione di ripetizione che mantiene il cursore in movimento fino a che non si libera il tasto.

[INST/DEL]

Se si batte il tasto **[INST/DEL],** il cursore si sposta di uno spazio, cancellando (DELeting-cancellazione) il carattere precedentemente battuto. Se ci si trova nel mezzo di una riga, il carattere alla sinistra viene cancellato ed i caratteri alla destra si spostano automaticamente per occuparne lo spazio.

Premendo il tasto **[INST/DEL]** preceduto dal tasto **[SHIFT]** è possibile inserire (INSerT-inserimento) informazioni su una riga. Per esempio, se ci si accorge di un errore di battuta all'inizio di una riga - probabilmente si è saltato una parte di un nome - si può usare il tasto **[CRSR LEFT/RIGHT]** per ritornare sull'errore e quindi battere **[INST/DEL]** per inserire uno spazio. Basta quindi battere la lettera mancante.

[CLR/HOME]

Il tasto **[CLR/HOME]** porta il cursore sulla posizione «HOME» ossia sulla posizione di partenza, che si trova nell'angolo superiore sinistro dello schermo. La pressione del tasto **[CLR/HOME]** preceduta dal tasto **[SHIFT]** cancella lo schermo e riporta il cursore in posizione di partenza.

[RESTORE]

Il tasto **[RESTORE]** (ripristino) funziona come implica lo stesso nome e cioè ripristina il computer alla condizione normale in cui si trovava prima di modificarla con un programma o qualche comando. Di questo tasto si parlerà a lungo nei prossimi capitoli.

I TASTI DI FUNZIONI

I quattro tasti posti sul lato destro della tastiera possono essere «programmati» per svolgere numerose funzioni. Essi possono cioè essere definiti in molti modi per svolgere i rispettivi compiti.

[IMMAGINE PAG 16]

[CTRL]

Il tasto **[CTRL]** che significa ConTRoL (controllo) consente di definire i colori e di eseguire altre funzioni specializzate. Per accedere ad una funzione di controllo basta tener abbassato il tasto **[CTRL]** e mentre si preme il tasto corrispondente alla funzione desiderata. Si è già avuto la possibilità di provare il funzionamento del tasto **[CTRL]** quando si sono cambiati i colori del testo per creare diverse barre colorate durante la procedura di messa a punto.

[RUN/STOP]

Normalmente la pressione del tasto **[RUN/STOP]** interrompe l'esecuzione di un programma BASIC. Esso segnala cioè al computer di interrompere (STOP) l'esecuzione di qualche cosa. L'uso del tasto **[RUN/STOP]** preceduto dal tasto **[SHIFT]** consente di caricare automaticamente un programma da nastro.

IL TASTO COMMODORE [C=]

Il tasto COMMODORE **[C=]** esegue numerose funzioni. Prima di tutto consente di passare sullo schermo dal modo testo al modo grafici.

All'accensione, il computer è predisposto nel modo maiuscole/grafici per cui tutto ciò che viene battuto comparirà in lettere maiuscole. Come si è già detto, l'uso del tasto **[SHIFT]** in questo modo visualizza il carattere grafico posto sul lato destro dei tasti.

Se si tiene abbassato il tasto **[C=]** e si preme contemporaneamente il tasto **[SHIFT]**, lo schermo passa ai caratteri maiuscoli e minuscoli. Ora, se si tiene abbassato il tasto **[C=]** e si preme qualsiasi altro tasto con un

simbolo grafico, compare il simbolo grafico posto sul lato sinistro del tasto.

Per ritornare al modo maiuscole/grafici, tenere abbassato il tasto **[C=]** e premere di nuovo **[SHIFT].**

La seconda funzione del tasto **[C=]** è di consentire l'accesso ad una seconda serie di otto colori di testo. Tenendo abbassato il tasto ed uno qualsiasi dei tasti numerici, qualsiasi testo sarà battuto nel colore alternativo disponibile in relazione al tasto abbassato. Il Capitolo 5 elenca i colori di testo disponibili ciascun tasto.

RITORNO AL FUNZIONAMENTO NORMALE

Ora che si è avuta la possibilità di osservare la tastiera, è possibile esplorare una delle molte capacità del COMMODORE 64.

Se sullo schermo figurano ancora le barre a colori create durante la regolazione del televisore, premere i tasti **[SHIFT] e [CLR/HOME].** Lo schermo dovrebbe cancellarsi ed il cursore disporsi in posizione di partenza (angolo superiore sinistro dello schermo).

Ora, premere simultaneamente **[C=]** ed il tasto **[7]**. La manovra riporta in azzurro il colore del testo. C'è però un'altra operazione richiesta per riportare tutto in condizioni normali. Occorre cioè tenere abbassato il tasto **[CTRL]** ed il tasto **[0]** (Attenzione: Zero e non la O maiuscola) per riportare lo schermo al funzionamento normale. Ci si ricorderà che era stato predisposto lo schermo in negativo con i tasti **[CTRL] [9]** per creare le barre colorate (le barre colorate erano in effetti degli spazi in negativo). Se lo schermo fosse stato predisposto nel modo normale durante la prova dei colori, il cursore si sarebbe spostato ma avrebbe lasciato spazi vuoti.

SUGGERIMENTO:

Dopo aver Imparato a fare le cose nel modo difficile, ecco un modo semplice per riportare il sistema al modo di funzionamento normale. Premere simultaneamente **[RUN/STOP] [RESTORE]** Ciò cancella lo schermo e riporta tutto al funzionamento normale. Se nel computer c'è un programma, questo viene lasciato inalterato. Vale la pena di ricordare questa sequenza, particolarmente quando si esegue molta programmazione. Se vi vuole ripristinare la macchina come avviene quando la si spegne e la si riaccende, battere: SYS 64738 e premere **[RETURN]**. Fare attenzione ad usare questo comando! Esso cancella qualsiasi programma o informazione che si trova correntemente nel computer.

CARICAMENTO E SALVATAGGIO DI PROGRAMMI

Una delle caratteristiche più importanti del COMMODORE 64 è la sua capacità di salvare e caricare programmi su e da cassetta di nastro o disco.

Questa capacità consente di salvare i programmi per utilizzarli in un tempo successivo o di acquistare programmi pronti da usare con il COMMODORE 64.

Assicurarsi che l'unità disco o l'unità datassette siano collegate correttamente.

Caricamento di programmi pronti

Per coloro che sono interessati ad usare soltanto programmi pronti disponibili su cartucce, cassette o disco ecco come procedere:

1. CARTUCCE: il computer COMMODORE 64 dispone di una serie di programmi e di giochi su cartuccia. I programmi offrono una grande varietà di applicazioni personali e gestionali ed i giochi sono quelli che si trovano normalmente nelle sale specializzate e non imitazioni. Per caricare questi giochi, accendere per prima cosa il televisore. Quindi SPEGNERE il COMMODORE 64. **OCCORRE SEMPRE SPEGNERE IL COMMODORE 64 PRIMA DI INSERIRE O RIMUOVERE LA CARTUCCIA ALTRIMENTI LA SI DISTRUGGE!** Successivamente inserire la cartuccia. Accendere ora il COMMODORE 64. Infine battere l'appropriato tasto START come indicato sul foglio di istruzione che viene fornito con ciascun gioco.

2. CASSETTE: Usare il registratore DATASETTE e le normali cassette audio che vengono fornite come parte del programma. Assicurarsi che il nastro sia completamente riavvolto all'inizio del primo lato. Quindi basta battere LOAD. Il computer risponde con PRESS PLAY ON TAPE cui occorre rispondere premendo il tasto PLAY sul datassette. A questo punto lo schermo del computer si cancella fino a che non viene trovato il programma. A questo punto sullo schermo comparirà FOUND (NOME DEL PROGRAMMA). Premere ora il tasto **[C=]**. L'operazione provoca il caricamento del programma nel computer. Se si vuol interrompere il caricamento basta premere il tasto **[RUN/STOP].**

3. DISCO: Usando l'unità disco, inserire delicatamente il disco pre-programmato in modo che la sua etichetta sia rivolta verso l'alto e verso l'operatore. Individuare la piccola tacca sul disco (potrebbe essere coperta con un piccolo pezzo di nastro). Se si inserisce il disco correttamente, la tacca si dovrebbe trovare sul lato sinistro. Una volta che il disco è all'interno chiudere lo sportello di protezione premendo sulla leva. Battere ora LOAD «NOME PROGRAMMA», 8 e battere quindi il tasto **[RETURN].** Il disco frullerà leggermente e sullo schermo comparirà:

```
SEARCHING FOR PROGRAM NAME
LOADING

READY
```

Quando compare READY ed il cursore, battere RUN. A questo punto il programma è pronto per l'uso.

Caricamento di programmi da nastro

Il caricamento di un programma da nastro o da disco è altrettanto semplice. Per caricare da nastro occorre riavvolgere il nastro all'inizio e battere:

```
LOAD "PROGRAM NAME"
```

Se non si ricorda il nome del programma, basta battere LOAD per caricare in memoria il primo programma sul nastro.

Dopo aver premuto **[RETURN]** il computer risponde con:

```
PRESS PLAY ON TAPE
```

Dopo aver premuto il tasto di riproduzione, lo schermo si cancella, lasciando soltanto il colore di fondo mentre il computer cerca il programma.
Una volta trovato, sullo schermo compare:

```
FOUND PROGRAM NAME
```

Per caricare (LOAD) il programma, premere il tasto **[C=]**. Per uscire dalla procedura caricamento, premere il tasto **[RUN/STOP].** Se si batte il tasto **[C=]**, lo schermo assume il colore del bordo mentre il programma viene caricato.
Al termine della procedura di caricamento, lo schermo ritorna alla condizione normale e compare la richiesta READY.

Caricamento di programmi da disco

Il caricamento di un programma da disco segue la stessa procedura.
Battere:

```
LOAD "PROGRAM NAME",8
```

Dopo aver battuto **[RETURN]** il disco inizia a frullare e sullo schermo compare:

```
SEARCHING FOR PROGRAM NAME
LOADING

READY
```

NOTA:
Quando si carica un nuovo programma nella memoria del computer, qualsiasi istruzione che si trovava precedentemente nel computer viene cancellata. Assicurarsi di salvare un programma sul quale si sta lavorando

prima di caricarne uno nuovo. Una volta che un programma è stato caricato, è possibile eseguirlo (RUN), listarlo (LIST) o effettuare modifiche e salvare la nuova versione.

Salvataggio di programmi su nastro

Dopo aver immesso un programma, se lo si vuole salvare su nastro, battere:

```
SAVE "PROGRAM NAME"
```

«PROGRAM NAME» (Nome programma) può essere una combinazione di un massimo di 16 caratteri. Dopo aver battuto il computer risponde con:

```
PRESS PLAY AND RECORD ON TAPE
```

Premere contemporaneamente i tasti di registrazione e di riproduzione sul datassette. Lo schermo si cancella, facendo comparire il colore del bordo.

Una volta che il programma è salvato su nastro, ricompare la richiesta READY indicando che è possibile iniziare a lavorare su un altro programma o spegnere il computer.

Salvataggio di programmi su disco

Il salvataggio di un programma su disco è ancora più semplice. Battere:

```
SAVE "PROGRAM NAME",8
```

L'8 è il codice del disco ed in questo modo si fa sapere al computer che si vuole salvare il programma sul disco.

Dopo aver premuto **[RETURN]**, il disco inizia a frullare ed il computer risponde con:

```
SAVING PROGRAM NAME
OK
READY
```

STAMPA E CALCOLI

Dopo aver imparato un paio delle operazioni più difficili necessarie per conservare i programmi di interesse, è possibile creare qualche programma da salvare e usare successivamente.

Cercare di battere quanto segue, esattamente come scritto:

```
PRINT "COMMODORE 64"        <--- Battere questa riga e premere [RETURN]
COMMODORE 64                <--- Battuto dal computer

READY
```

Se si fa un errore di battitura, usare il tasto **[INST/DEL]** per cancellare il carattere immediatamente alla sinistra del cursore. È possibile cancellare quanti caratteri è necessario.

Vediamo ora cosa è successo nell'esempio. Innanzitutto si è istruito (comandato) il computer a PRINT (battere ossia scrivere) qualsiasi cosa si trovava all'interno delle virgolette. Battendo **[RETURN]** si è detto al computer di fare ciò che si era indicato e sullo schermo è comparsa la scritta COMMODORE 64.

Quando si usa l'istruzione PRINT in questa forma, tutto ciò che è racchiuso tra virgolette viene stampato ossia scritto esattamente come lo si è battuto.

Se il computer risponde con:

```
?SYNTAX ERROR
```

chiedersi se non si è per caso fatto un errore di battitura o si sono dimenticate le virgolette. Il computer è preciso e si aspetta che le istruzioni vengano date in una forma specifica altrettanto precisa.

Ma non è il caso di preoccuparsi: basta ricordarsi di immettere le cose come vengono presentate negli

esempi. Ricordarsi che non è possibile danneggiare il computer qualunque cosa si batta su di esso e che il modo migliore per imparare il BASIC è di provare varie e vedere cosa succede.

PRINT è uno dei comandi più utili e più potenti nel linguaggio BASIC. Con esso è possibile visualizzare pressoché qualsiasi cosa si desideri, compresi i grafici ed i risultati dei calcoli.

Per esempio, provare quanto segue. Cancellare lo schermo tenendo abbassato il tasto **[SHIFT]** ed il tasto **[CLR/HOME]** e battere (assicurarsi di usare il tasto «1» e non la lettera «1»):

```
PRINT 12 + 12   <--- Battere queste righe e premere [RETURN]
24              <--- Il computer ha battuto il risultato

READY
```

Si è così scoperto che il COMMODORE 64, nella sua forma base, è un calcolatore. Il risultato «24» è stato calcolato e stampato automaticamente. In effetti è inoltre possibile eseguire sottrazioni, moltiplicazioni, divisioni, elevamento ad esponente e funzioni matematiche avanzate tipo calcolo di radici quadrate, ecc. E non si è limitati ad un singolo calcolo su una riga, ma di questo parlerà più avanti.

Notare che nella forma suddetta, PRINT si è comportato in maniera diversa dal primo esempio. In questo caso, viene battuto un valore o un risultato di un calcolo anziché il messaggio esatto che era stato immesso, in quanto sono state omesse le virgolette.

Somma

Il segno più (+) indica l'addizione: si istruisce cioè il computer a stampare il risultato di 12 sommato a 12. Altre operazioni aritmetiche assumono una forma simile all'addizione. Ricordarsi di battere sempre **[RETURN]** dopo aver battuto PRINT ed il calcolo da eseguire.

Sottrazione

Per sottrarre, usare il segno meno (-) convenzionale. Battere:

```
PRINT 12 - 9    <--- Battere [RETURN]
3
```

Moltiplicazione

Il segno di moltiplicazione è l'asterisco (*). Se si vuole moltiplicare 12 volte 12, si batterà:

```
PRINT 12 * 12   <--- Battere [RETURN]
144
```

Divisione

La divisione è indicata dalla familiare barretta «/». Ad esempio per dividere 144 per 12, battere:

```
PRINT 144/12    <--- Battere [RETURN]
12
```

Elevamento ad esponente

In maniera analoga è possibile facilmente elevare un numero ad una potenza (il che equivale a moltiplicare un numero per sé stesso per un numero specificato di volte). La freccia verso l'alto «↑» significa elevamento ad esponente.

```
PRINT 12 ^ 5
248832
```

Ciò equivale a battere:

```
PRINT 12 * 12 * 12 * 12 * 12
248832
```

SUGGERIMENTO:

Il BASIC dispone dl un certo numero dl scorciatoie per fare determinate cose. Una di esse è l'abbreviazione dei comandi (o parole chiave) BASIC. Ad esempio, può essere usato un ? in luogo di PRINT. Man mano si procede verranno presentati molti comandi. L'Appendice D mostra l'abbreviazione per ciascuno di essi e ciò che compare sullo schermo quando si batte la forma abbreviata.

L'ultimo esempio richiama un punto molto importante: è possibile eseguire molti calcoli sulla stessa riga e questi calcoli possono essere di tipo misto.

È possibile calcolare questo problema:

```
+--- Questo ? sostituisce la parola PRINT
|
? 3 + 5 -7 +2
3
```

Fino a questo punto si sono usati numeri piccoli ed esempi semplici, ma COMMODORE 64 è in grado di eseguire calcoli più complessi.

Si potrebbero ad esempio sommare numerose grandi cifre. Provare con questo esempio, senza però usare le virgole, altrimenti si ottiene un errore:

```
? 123.45 + 345.78 + 7895.687
8364.917
```

Provare ora questo:

```
? 12123123.45 + 345.78 + 7895.687
12131364.9
```

Se si prende il gusto di sommare manualmente, si trova un diverso risultato.

Cosa è successo? Quantunque il computer abbia molta potenza, ha tuttavia un limite ai numeri che gestire. Il COMMODORE 64 può lavorare con numeri che contengono 10 cifre, ma quando un numero viene stampato, vengono visualizzate soltanto 9 cifre.

Così nell'esempio, il risultato è stato «arrotondato» per inserirsi nel campo appropriato. COMMODORE 64 arrotonda per eccesso quando la successiva cifra è cinque o più, arrotonda per difetto quando la successiva cifra è quattro o meno.

I numeri compresi fra 0.01 e 999,999,999 sono stampati usando la notazione standard. I numeri al di fuori di questo campo vengono stampati usando la notazione scientifica.

La notazione scientifica non è che un altro modo per esprimere un numero molto grande o molto piccolo sotto forma di potenza di 10.

Se si batte:

```
? 123000000000000000
1.23E+17
```

Ciò equivale a dire 1.23 * 10↑17 e si usa per rendere le cose molto semplici.

C'è però un limite ai numeri che il computer può gestire anche nella notazione scientifica. Questi limiti sono:

Numero più grande: ± 1.70141183E+38
Numero più piccolo: ± 2.93873588E-39

PRECEDENZA

Se si cercasse di eseguire qualche calcolo misto diverso dagli esempi mostrati precedentemente, si potrebbero non trovare i risultati previsti. Il motivo è che il computer esegue i calcoli in un certo ordine.

In questo calcolo:

20 + 8/2

non è possibile dire se la risposta deve essere 24 o 14 fino a che non si sa in quale ordine si eseguono i calcoli. Se si somma 20 a 8 diviso per 2 (ossia 4), il risultato è 24. Ma se si somma 20 a 8 e si divide quindi per 2 il risultato è 14. Provare con l'esempio e vedere cosa si ottiene come risultato.

Il motivo per aver ottenuto 24 è che il COMMODORE 64 esegue i calcoli da sinistra a destra secondo quanto segue:

Primo:	-	segno meno che indica i numeri negativi
Secondo:	^	elevamento ad esponente da sinistra a destra
Terzo:	* /	moltiplicazione e divisione, da sinistra a destra
Quarto:	+ -	addizione e sottrazione, da sinistra a destra

Seguendo l'ordine di precedenza indicato nel suddetto esempio la divisione viene eseguita per prima e quindi viene eseguita l'addizione per dare un risultato di 24.

È bene a questo punto esercitarsi con alcuni problemi propri e prevedere i risultati secondo le regole sopra indicate.

C'è anche un modo facile per variare la precedenza, usando cioè le parentesi per separare le operazioni che si vogliono eseguire per prime.

Per esempio, se si vuole dividere 35 per 5 più 2 si batte:

```
? 35 / 5 + 2
9
```

e si ottiene 35 diviso 5 con 2 aggiunto al risultato, il che non è ciò che si intendeva. Provare invece in questo modo:

```
? 35 / (5 + 2)
5
```

È successo che il computer valuta per primo ciò che è contenuto nelle parentesi. Se ci sono parentesi all'interno di altre parentesi, vengono calcolate per prime le parentesi più interne.

Dove ci sono numerose parentesi su una riga, ad esempio:

```
? (12 + 9) * (6 + 1)
147
```

il computer le valuta da sinistra a destra. Qui 21 verrebbe moltiplicato per 7 per avere il risultato di 147.

COME COMBINARE LE COSE

Quantunque sia stato dedicato un po' di tempo ad argomenti che potrebbero sembrare non importanti, i particolari qui presentati avranno tuttavia più senso una volta iniziata la programmazione e si dimostreranno preziosi.

Per dare un'idea di come le cose vadano al loro posto, si consideri quanto segue: come è possibile combinare tutti e due i tipi di istruzioni di stampa finora esaminate per stampare qualcosa che abbia più significato sullo schermo?

Si sa che racchiudendo qualcosa tra le virgolette si stampano quelle informazioni sullo schermo esattamente come sono state immesse e che usando gli operatori matematici è possibile eseguire calcoli. Così perché non combinare i due tipi di istruzione PRINT come segue:

```
                +--- Il punto e virgola indica assenza di spazio
                |
? "5 * 9 = "; 5 * 9
5 * 9 = 45
```

Quantunque ciò possa sembrare in un certo modo ridondante, non si è fatto altro che usare insieme entrambi i tipi di istruzione PRINT. La prima parte stampa «5 * 9 =» esattamente come è stato battuto. La seconda parte esegue il lavoro effettivo e stampa il risultato, con il punto e virgola che separa la parte messaggio dell'istruzione dal calcolo effettivo.

Occorre sempre separare le parti di un'istruzione di stampa mista con qualche segno di punteggiatura perché le cose funzionino correttamente. Provare con una virgola in luogo del punto e virgola e vedere cosa succede.

Per il curioso, il punto e virgola può far sì che la successiva parte dell'istruzione venga stampata immediatamente dopo la parte precedente, senza spazi. La virgola fa qualcosa di diverso. Quantunque sia un separatore accettabile, distanzia le cose ulteriormente. Se si batte:

```
? 2,3,4,5,6     <--- Battere [RETURN]
2         3           4           5
6
```

i numeri vengono stampati occupando tutto lo schermo ed anche la riga successiva.

Lo schermo del COMMODORE 64 è suddiviso in quattro aree di 10 colonne ciascuna. La virgola esegue la tabulazione di ciascun risultato nella successiva area disponibile. Dato che è stato chiesto di stampare più informazioni di quante ne possa accogliere la riga, (si è cioè tentato di inserire cinque aree di 10 colonne su una riga), l'ultima voce è stata spostata alla riga successiva.

La differenza base tra la virgola ed il punto e virgola nella formattazione delle istruzioni PRINT può essere usata vantaggiosamente creando schermi più complessi: essa consente di creare molto facilmente alcuni risultati sofisticati.

CAPITOLO 3

INIZIO DELLA PROGRAMMAZIONE IN BASIC

- La fase successiva GOTO
- Suggerimenti per la correzione
- Variabili
- IF...THEN
- Le iterazioni FOR...NEXT

LA FASE SUCCESSIVA

Finora sono state eseguite semplici operazioni immettendo nel computer una singola riga di istruzioni. Una volta abbassato il tasto **[RETURN]**, l'operazione specificata viene eseguita immediatamente. Questo modo di procedere è detto modo CALCOLATORE o IMMEDIATO.

Ma per effettuare qualche cosa di importante, occorre far sì che il computer operi con più di un'istruzione su una sola riga. Un certo numero di istruzioni combinate fra di loro è detto PROGRAMMA e consente di usare la piena potenza del COMMODORE 64.

Per rendersi come sia facile scrivere il primo programma COMMODORE 64, provare come segue:

Cancellare lo schermo tenendo abbassato il tasto **[SHIFT]** e quindi abbassando il tasto **[CLR/HOME]**.

Battere NEW e premere **[RETURN]**. (Ciò cancella qualsiasi numero che potrebbe essere rimasto nel

computer dalle precedenti sperimentazioni).

Battere ora quanto segue esattamente come indicato (ricordarsi di battere **[RETURN]** dopo ciascuna riga).

```
10 ?"COMMODORE 64"
20 GOTO 10
```

Battere ora RUN e premere ancora **[RETURN]** ed osservare ciò che succede. Lo schermo si anima facendo comparire la scritta COMMODORE 64. Dopo aver osservato lo schermo, premere **[RUN/STOP]** per interrompere il programma.

In questo breve programma sono stati introdotti numerosi ed importanti concetti che formano la base di tutta la programmazione.

Notare che si è fatta precedere ciascuna istruzione da un numero, detto numero di RIGA, numero che dice al computer in quale ordine lavorare per ciascuna istruzione. Questi numeri rappresentano anche un punto di riferimento in caso in cui un programma debba tornare indietro ad una particolare riga. I numeri di riga possono essere rappresentati da qualsiasi valore intero compreso tra 0 e 63.999.

```
10 PRINT "COMMODORE 64"
 |   |
 |   +---- ISTRUZIONE
 |
 +------- NUMERO DI LINEA
```

```
COMMODORE 64
COMMODORE 64
COMMODORE 64
COMMODORE 64
COMMODORE 64
COMMODORE 64
COMMODORE 64
COMMODORE 64
COMMODORE 64
COMMODORE 64
COMMODORE 64
COMMODORE 64
BREAK IN 10
READY
```

È buona prassi di programmazione numerare le righe ad incrementi di 10 nel caso in occorresse inserire successivamente altre istruzioni.

Oltre a PRINT, il programma ha usato anche un altro comando BASIC, e cioè GOTO. Questo comando istruisce il computer a portarsi direttamente su una particolare riga, ad eseguirla e quindi a continuare da quel punto.

```
+--> 10 PRINT "COMMODORE 64"
|
+--- 20 GOTO 10
```

Nell'esempio, il programma stampa il messaggio nella riga 10, va alla riga successiva (20), che dà istruzioni di ritornare alla riga 10 e stampa il messaggio di nuovo. Quindi il ciclo si ripete. Dato che non si è dato al computer un modo per uscire da questo circolo vizioso, il programma continuerà all'infinito o fino a che non lo si interrompe fisicamente con il tasto **[RUN/STOP]**.

Una volta interrotto il programma battere: LIST. Il programma sarà visualizzato intatto, in quanto è ancora nella memoria del computer. Notare anche che il computer ha convertito automaticamente il punto di domanda in PRINT. Il programma può essere ora cambiato, salvato o eseguito di nuovo.

Un'altra importante differenza tra battere qualche cosa nel modo immediato e scrivere un programma è che una volta eseguita l'istruzione e cancellato lo schermo, l'istruzione immediata va persa. Per contro è sempre possibile far ricomparire un programma semplicemente battendo LIST.

Fra l'altro, parlando di abbreviazioni, non dimenticare che il computer può esaurire lo spazio su una riga se se ne usano troppe.

SUGGERIMENTI PER LE CORREZIONI

Se si compie un errore sulla riga, ci sono numerose possibilità di correzione.

1. È possibile ribattere una riga in qualsiasi momento: il computer la sostituirà automaticamente alla vecchia.
2. Una riga indesiderata può essere cancellata semplicemente battendo il numero di riga e **[RETURN]**.
3. È possibile inoltre correggere facilmente una riga esistente usando i tasti de; cursore ed i tasti di editing.

Si supponga di aver compiuto un errore di battitura in una riga dell'esempio. Per correggerlo senza ribattere l'intera riga procedere come segue:

Battere LIST, quindi usando insieme i tasti **[SHIFT]** e **[CRSR UP/DOWN]** spostare il cursore verso l'alto fino a che non è posizionato sulla riga che deve essere modificata.

Ora usare il tasto del cursore con freccia verso destra per spostare il cursore sul carattere che si vuole modificare, ribattendo la modifica sul vecchio carattere. Premere infine **[RETURN]** per far sì che la riga corretta sostituisca la vecchia.

Se occorre più spazio sulla riga, posizionare il cursore nel punto in cui occorre spazio e premere contemporaneamente i tasti **[SHIFT]** e **[INST/DEL]**.

Con questa manovra si apre uno spazio. Battere ora le ulteriori informazioni e premere **[RETURN]** Analogamente è possibile cancellare caratteri indesiderati disponendo il cursore alla destra del carattere indesiderato e premendo il tasto **[INST/DEL]**.

Per verificare che le modifiche siano state immesse, battere di nuovo LIST per far ricomparire il programma corretto. Le righe non devono essere necessariamente immesse in ordine numerico: pensa il computer ad inserirle automaticamente nell'appropriata sequenza.

Provare a correggere il programma esemplificativo cambiando la riga 10 ed aggiungendo un punto e virgola alla fine della riga come indicato a pagina 35. Quindi eseguire (RUN) di nuovo il programma.

```
10 PRINT"COMMODORE";        Non dimenticare di spostare
                            il cursore oltre la riga 20
                            prima di eseguire il programma
```

VARIABILI

Le variabili sono alcune delle caratteristiche più usate di qualsiasi linguaggio di programmazione in quanto possono rappresentare nel computer un numero molto maggiore di informazioni. La conoscenza del modo di funzionamento delle variabili rende il calcolo più facile e consente di eseguire operazioni non altrimenti possibili.

```
COMMODORE     COMMODORE     COMMODORE     COMMODORE
COMMODORE     COMMODORE     COMMODORE     COMMODORE
COMMODORE     COMMODORE     COMMODORE     COMMODORE
COMMODORE     COMMODORE     COMMODORE     COMMODORE
COMMODORE     COMMODORE     COMMODORE     COMMODORE
COMMODORE     COMMODORE     COMMODORE     COMMODORE
COMMODORE     COMMODORE     COMMODORE     COMMODORE
COMMODORE     COMMODORE     COMMODORE     COMMODORE
COMMODORE     COMMODORE     COMMODORE     COMMODORE
COMMODORE     COMMODORE     COMMODORE     COMMODORE
COMMODORE     COMMODORE     COMMODORE     COMMODORE
COMMODORE     COMMODORE     COMMODORE     COMMODORE
COMMODORE     COMMODORE     COMMODORE     COMMODORE
BREAK IN 10
READY
```

Si immagini di avere un certo numero di scatole all'interno del computer, ciascuna delle quali contiene un numero o una stringa di caratteri di testo. Ciascuna di queste scatole deve essere contrassegnata con un nome a scelta. Questo nome è detto variabile e rappresenta le informazioni contenute nella rispettiva scatola.

Per esempio dicendo:

```
10 X% = 15
20 X = 23.5
30 X$ = "LA SOMMA DI X%+X = "
```

Il computer può rappresentare le variabili in questo modo:

```
X%  15
X   23.5
X$  LA SOMMA DI X%+X =
```

Una nuova variabile rappresenta la scatola o la locazione di memoria in cui viene caricato il valore corrente della variabile. Come è possibile vedere, ad una variabile si può assegnare un numero intero o un numero in virgola mobile o una stringa di testo.

Il simbolo % che segue un nome variabile indica che la variabile rappresenterà un numero intero. Quelli che seguono sono nomi di variabili valide intere:

A%
X%
A1%
NM%

Il «$» che segue il nome variabile indica che la variabile rappresenta una stringa di testo. Quelli che seguono sono esempi di variabili stringa:

A$
X$
MI$

Le variabili in virgola mobile seguono lo stesso formato, con l'indicatore di tipo:

Al
X
Y
Ml

Nell'assegnare un nome ad una variabile ci sono alcune cose da tener presenti. Innanzitutto una variabile può avere uno o due caratteri. Il primo carattere deve essere un carattere alfabetico da A a Z; il secondo carattere può essere un carattere alfabetico o numerico (nel campo da O a 9). Può essere incluso un terzo carattere per indicare il tipo di variabile (intera o stringa di testo), % o $.

È possibile usare nomi variabili con più di due caratteri alfabetici ma solo i primi due vengono riconosciuti dal computer. Così PA e PARTNO sono identici e fanno riferimento alla stessa variabile.

L'ultima regola per i nomi variabili è semplice: essi non possono contenere qualsiasi parola chiave BASIC (parole riservate) tipo GOTO, RUN, ecc. Fare riferimento all'Appendice D per un elenco completo delle parole riservate BASIC.

Per vedere come si possono utilizzare le variabili, battere il programma completo introdotto precedentemente ed eseguirlo (RUN) Ricordarsi di premere **[RETURN]** dopo ciascuna riga del programma.

vedere, ad una variabile si può assegnare un numero intero o un numero in virgola mobile o una stringa di testo.

Il simbolo % che segue un nome variabile indica che la variabile rappresenterà un numero intero. Quelli che seguono sono nomi di variabili valide intere:

A%
X%
A1%

NM%

Il «$» che segue il nome variabile indica che la variabile rappresenta una stringa di testo. Quelli che seguono sono esempi di variabili stringa:

A$
X$
MI$

Le variabili in virgola mobile seguono lo stesso formato, con l'indicatore di tipo:

Al
X
Y
Ml

Nell'assegnare un nome ad una variabile ci sono alcune cose da tener presenti. Innanzitutto una variabile può avere uno o due caratteri. Il primo carattere deve essere un carattere alfabetico da A a Z; il secondo carattere può essere un carattere alfabetico o numerico (nel campo da O a 9). Può essere incluso un terzo carattere per indicare il tipo di variabile (intera o stringa di testo), % o $.

È possibile usare nomi variabili con più di due caratteri alfabetici ma solo i primi due vengono riconosciuti dal computer. Così PA e PARTNO sono identici e fanno riferimento alla stessa variabile.

L'ultima regola per i nomi variabili è semplice: essi non possono contenere qualsiasi parola chiave BASIC (parole riservate) tipo GOTO, RUN, ecc. Fare riferimento all'Appendice D per un elenco completo delle parole riservate BASIC.

Per vedere come si possono utilizzare le variabili, battere il programma completo introdotto precedentemente ed eseguirlo (RUN) Ricordarsi di premere **[RETURN]** dopo ciascuna riga del programma.

```
NEW
10 X% = 15
20 X = 23.5
30 X$ = "THE SUM OF X% + X = "
40 PRINT "X% = "; X%, "X = "; X
50 PRINT X$; X% + X
```

Se tutto è stato eseguito come indicato, si dovrebbe ottenere sullo schermo il seguente risultato.

```
RUN
X% = 15    X = 23.5
THE SUM OF X% + X = 38.5
READY
```

Sono stati riuniti tutti i trucchi finora imparati per formattare uno schermo così come lo si vede e per stampare la somma delle due variabili.

Nelle righe 10 e 20 è stato assegnato un valore intero a X% ed assegnato un valore in virgola mobile a X. Ciò inserisce il numero associato con la variabile nella rispettiva «scatola». Nella riga 30 è stata assegnata una stringa di testo a X$. La riga 40 riunisce i due tipi di istruzione PRINT per battere un messaggio ed il valore effettivo di X% e di X. La riga 50 stampa la stringa di testo assegnata a X$ e la somma di X% e X.

Notare che quantunque venga usato X come parte di ciascun nome variabile, gli identificatori % e $ rendono X%, X e X$ unici, e cioè fanno in modo che essi rappresentino tre variabili distinte.

Ma le variabili sono molte più potenti. Se se ne cambia il valore, il nuovo valore sostituisce quello originale nella stessa «scatola». Ciò consente di scrivere un'istruzione del tipo:

X = X + 1

L'istruzione di questo genere non verrebbe mai accettata nell'algebra normale ma rappresenta per contro uno dei concetti più usati nella programmazione e significa «prendere il valore corrente di X, aggiungere uno ed

inserire la nuova somma nella «scatola» che rappresenta X».

IF...THEN

Armati della capacità di aggiornare facilmente il valore delle variabili, è possibile ora provare un programma di questo tipo:

```
NEW
10 CT = 0
20 ?"COMMODORE 64"
30 CT = CT + 1
40 IF CT < 5 THEN 20
50 END
RUN
COMMODORE 64
COMMODORE 64
COMMODORE 64
COMMODORE 64
COMMODORE 64
```

Non si è fatto altro che introdurre due nuovi comandi BASIC e disporre qualche controllo sul breve programma di stampa introdotto all'inizio del capitolo.

IF...THEN aggiunge una certa logica al programma. Esso dice che se (IF) una condizione è vera, allora (THEN) occorre fare qualche cosa. Se (IF) la condizione non è più vera, allora (THEN) occorre passare alla successiva riga nel programma.

È possibile fissare un certo numero di condizioni usando l'istruzione IF...THEN:

SYMBOLO	*SIGNIFICATO*
<	Minore di
>	Maggiore di
=	Uguale a
<>	Diverso da
>=	Maggiore di o uguale a
<=	Minore di o uguale a

L'uso di una qualsiasi di queste condizioni è facile ma sorprendentemente efficace.

```
    10 CT = 0
+-> 20 ?"COMMODORE 64"
|   30 CT = CT + 1
+-< 40 IF CT < 5 THEN 20
     |
    50 END
```

Nel programma campione è stato creato un «loop» o «iterazione» sul quale sono stati posti alcuni vincoli dicendo: Se (IF) un valore è minore di un certo numero allora (THEN) occorre fare qualche cosa.

La riga 10 definisce CT (CounT-Conteggio) uguale a 0. La riga 20 stampa il messaggio. La riga 30 aggiunge uno alla variabile CT. Questa riga conta quante volte occorre eseguire il loop o iterazione. Ogni volta che l'iterazione viene eseguita, CT aumenta di uno.

La riga 40 è la riga di controllo. Se CT è meno di 5, indicando che è stata eseguita l'iterazione cinque volte, il programma ritorna alla riga 20 e stampa di nuovo. Quando CT è uguale a 5 - indicando che è stato stampato cinque volte COMMODORE 64 - il programma va alla riga 50, che segnala la fine (END) del programma.

Provare praticamente il programma. Cambiando il limite CT nella riga 40 si può far stampare qualsiasi numero di righe.

IF...THEN ha molti altri usi, che verranno esaminati negli esempi successivi.

ITERAZIONI FOR...NEXT

C'è un modo più semplice e preferibile per eseguire ciò che è stato fatto nell'esempio precedente usando un'iterazione FOR...NEXT. Si consideri quanto segue:

```
NEW

10 FOR CT = 1 TO 5
20 PRINT "COMMODORE 64"
30 NEXT CT

RUN
COMMODORE 64
COMMODORE 64
COMMODORE 64
COMMODORE 64
COMMODORE 64
```

Come si può vedere, il programma se è ridotto notevolmente ed è più diretto.

CT inizia in i alla riga 10. Quindi la riga 20 esegue la stampa. Alla riga 30 CT è incrementato di 1. L'istruzione NEXT nella riga 30 rimanda automaticamente il programma alla riga 10 dove è 10 disposta la parte FOR dell'istruzione FOR...NEXT Questo processo continua fino a che CT non raggiunge il limite impostato.

La variabile usata in un'iterazione FOR...NEXT può essere incrementata di quantitativi più piccoli di 1, se necessario.

Provare con questo esempio:

```
NEW

10 FOR NB = 1 TO 10 STEP .5
20 PRINT NB,
30 NEXT NB

RUN
1         1.5         2.          2.5
3         3.5         4.          4.5
5         5.5         6           6.5
7         7.5         8           8.5
9         9.5         10
```

Se si immette e si esegue questo programma, si vedono comparire sullo schermo i numeri da 1 a 10 incrementi di 0,5.

Non si è fatto altro che stampare i valori che NB assume man mano che procede con le iterazioni.

È possibile anche specificare se la variabile debba aumentare o diminuire. Si sostituisca la seguente alla riga 10:

```
10 FOR NB = 10 to 1 STEP -.5
```

ed osservare che si verifica il contrario, dato che NB va da 10 a 1 in ordine discendente.

CAPITOLO 4

BASIC AVANZATO

- Introduzione
- Semplice esempio di animazione
  - Iterazione nidificata
- INPUT
- GET
- Numeri Casuali ed altre funzioni
- Gioco degli indovinelli
- Il lancio di dadi
- Grafici casuali
  - Funzioni CHR$ e ASC

INTRODUZIONE

I successivi capitoli sono stati scritti per coloro che hanno già una certa familiarità con il linguaggio di programmazione BASIC e con i concetti necessari per scrivere i programmi più avanzati.

Coloro invece che affrontano ora per la prima volta la programmazione, potranno trovare alcune informazioni leggermente troppo tecniche per comprendere completamente. Ma coraggio... poiché per i successivi Capitoli 6 e 7 (rispettivamente **GRAFICI ANIMATI e CREAZIONE DEL SUONO**) sono stati creati sette esempi scritti appositamente per i principianti. Gli esempi daranno una buona idea del modo in cui usare le sofisticate capacità sonore e grafiche disponibili sul COMMODORE 64.

Se si vuole saperne di più sulla compilazione di programmi in BASIC, consultare l'Appendice N che contiene una bibliografica adatta allo scopo.

Per chi ha già familiarità con la programmazione BASIC, questi capitoli aiuteranno ad affrontare le tecniche di programmazione avanzata BASIC. Informazioni più dettagliate si possono trovare nel **COMMODORE 64 PROGRAMMER'S REFERENCE MANUAL,** disponibile presso il Rivenditore COMMODORE.

SEMPLICI ESEMPIO DI ANIMAZIONE

È possibile provare qualcuna delle capacità grafiche di COMMODORE 64 riunendo ciò che è stato visto finora unitamente a qualche nuovo concetto. Chi è ambizioso può battere il seguente programma e vedere cosa succede. Si noterà che all'interno dell'istruzione di stampa è possibile includere comandi del cursore e comandi dello schermo. Quando in un listato di programma si vede comparire qualcosa del tipo (CRSR LEFT), tenere abbassato il tasto **[SHIFT]** e battere il tasto che sposta il cursore verso destra o verso sinistra. Lo schermo mostrerà la rappresentazione grafica di un cursore con treccia verso sinistra (due barre verticali invertite). Allo stesso modo, premendo **[SHIFT]** e **[CLR/HOME]** compare un cuore in negativo.

```
NEW
10 REM BOUNCING BALL
20 PRINT "{CLEAR}"
25 FOR X = 1 TO 10 : PRINT "{DOWN}" : NEXT
30 FOR BL = 1 TO 39
40 PRINT" {209}{LEFT}";:REM {CLEAR} IS A SHIFT-Q"
50 FOR TM = 1 TO 5
60 NEXT TM
70 NEXT BL
75 REM MOVE BALL RIGHT TO LEFT
80 FOR BL = 39 TO 1 STEP -1
90 PRINT" {LEFT*2}{209}{LEFT}";
100 FOR TM = 1 TO 5
110 NEXT TM
120 NEXT BL
130 GOTO 20
```

SUGGERIMENTO:

Tutte le parole in queste testo saranno completate su una riga. In ogni caso fino a che non si preme **[RETURN]** il COMMODORE 64 si sposterà automaticamente alla riga successiva anche nel mezzo di una parola.

Il programma presenterà una pallina rimbalzante che si muove sullo schermo da sinistra verso destra e da destra verso sinistra.

Se si osserva il programma da vicino (indicato a pagina 44) è possibile vedere come questa azione viene ottenuta.

La riga 10 è un REMark (nota) che spiega ciò che fa il programma; essa non ha però alcun effetto sul programma stesso.

```
        10 REM BOUNCING BALL
+-----> 20 PRINT "{CLEAR}"
| +---> 25 FOR X = 1 TO 10 : PRINT "{DOWN}" : NEXT
| |     30 FOR BL = 1 TO 39
| |     40 PRINT" {209}{LEFT}";:REM {CLEAR} IS A SHIFT-Q"
| | +-> 50 FOR TM = 1 TO 5
| | +-< 60 NEXT TM
| +---< 70 NEXT BL
|       75 REM MOVE BALL RIGHT TO LEFT
| +---> 80 FOR BL = 39 TO 1 STEP -1
| |     90 PRINT" {LEFT*2}{209}{LEFT}";
| | +-> 100 FOR TM = 1 TO 5
| | +-< 110 NEXT TM
| +---< 120 NEXT BL
+-----> 130 GOTO 20
```

La riga 20 cancella qualsiasi informazione dallo schermo.

La riga 25 PRINT (stampa) 10 comandi di movimento verso il basso del cursore. Ciò non fa che posizionare la pallina al centro dello schermo. Se la riga 25 fosse eliminata, la pallina si sposterebbe alla riga superiore dello schermo.

La riga 30 crea un'iterazione per spostare la pallina di 40 colonne da sinistra verso destra.

La riga 40 fa un mucchio di cose. Per prima cosa stampa uno spazio per cancellare le precedenti posizioni della pallina quindi stampa la pallina ed infine esegue un movimento verso sinistra del cursore per predisporre tutto cancellare di nuovo la posizione corrente della pallina.

L'iterazione creata nelle righe 50 e 60 rallenta leggermente la pallina ossia ritarda il programma. Senza di essa, la pallina si muoverebbe troppo velocemente per poterla vedere.

La riga 70 completa l'iterazione che stampa le palline sullo schermo, impostata nella riga 30. Ogni volta che l'iterazione viene eseguita, la pallina si muove di un altro spazio verso destra. Come notato dall'illustrazione, è stata inserita un'iterazione nell'ambito di un'altra.

Ciò è perfettamente accettabile. Si hanno dei problemi soltanto quando le iterazioni si incrociano una con l'altra. Nel compilare programmi è utile controllare come illustrato in questo caso, per assicurarsi che la logica di un'iterazione sia corretta.

Per vedere ciò che succederebbe incrociando un'iterazione, basta invertire le istruzioni delle righe 60 e 70. Si ottiene in tal caso un errore in quanto il computer si confonde e non può rendersi conto di quanto sta succedendo.

Le righe da 80 a 120 non fanno altro che invertire le fasi nella prima parte del programma e spostare la pallina da sinistra a destra verso sinistra. La riga 90 è leggermente diversa dalla riga 40 in quanto la pallina si muove in direzione opposta (occorre cancellare la pallina verso destra e spostarla verso sinistra).

E quando ciò è stato fatto il programma ritorna alla riga 20 per iniziare da capo l'intero processo. Interessante, non è vero?

Per una variazione sul programma correggere la riga 40 come segue:

40 PRINT "•" <--- Per creare lo •, tenere abbassato il tasto SHIFT e battere la lettera «Q»

Eseguire il programma e vedere cosa succede ora. Poiché è stato escluso il controllo del cursore, ciascuna pallina rimane sullo schermo fino a che non viene cancellata dal movimento della pallina da destra verso sinistra nella seconda parte del programma.

INPUT

Finora, tutto ciò che compariva in un programma era stato impostato prima della sua esecuzione. Una volta iniziato il programma nulla poteva essere cambiato. INPUT consente invece di trasmettere nuove informazioni ad un programma mentre è in esecuzione e farlo operare sulle nuove informazioni.
Per avere un'idea di come funziona INPUT, battere NEW **[RETURN]** ed immettere questo breve programma:

```
10 INPUT A$
20 PRINT "YOU TYPED: "A$
30 PRINT
40 GOTO 10
RUN
? COMMODORE 64               <--- Ciò che è stato battuto
YOU TYPED COMMODORE 64       <--- Il computer ha risposto così
```

Ciò che succede quando si esegue questo programma è molto semplice. Compare un punto di domanda per indicare che il computer è in attesa che si batta qualche cosa. Immettere qualsiasi carattere o gruppo di caratteri da tastiera e premere **[RETURN].** Il computer risponde con «YOU TYPED:» seguito dalle informazioni immesse.

Ciò può sembrare molto elementare ma si immagini cosa è possibile far fare al computer con le informazioni immesse.
È possibile INPUT (immettere) variabili numeriche o stringhe ed addirittura fare in modo che l'istruzione INPUT informi l'utente con un messaggio. il formato di INPUT è:

```
INPUT "MESSAGGIO RICHIESTA";Variabile
                |
                +--  La richiesta deve contenere meno di 40 caratteri
```

Oppure semplicemente:

INPUT VARIABILE

NOTA Per uscire da questo programma tenere abbassati i tasti **[RUN/STOP]** e **[RESTORE]**.

Il programma seguente non è soltanto utile ma dimostra molto di ciò che è stato finora presentato, compresa la nuova istruzione di input.

```
NEW

1 REM TEMPERATURE CONVERSION PROGRAM
5 PRINT "{CLR/HOME}"
10 PRINT "CONVERT FROM FAHRENHEIT OR CELSIUS (F/C): ": INPUT A$
20 IF A$ = "" THEN 10  <--- Nessuno spazio tra " e "
30 IF A$ = "F" THEN 100
40 IF A$ <> "C" THEN 10
50 INPUT "ENTER DEGREES CELSIUS: ";C
60 F = (C*9)/5+32
70 PRINT C;" DEG. CELSIUS ="; F ;"DEG. FAHRENHEIT"
80 PRINT
90 GOTO 10
100 INPUT "ENTER DEGREES FAHRENHEIT: ";F  <--- Non dimenticare di premere RETURN
110 C = (F-32)*/5/9
```

```
120 PRINT F;" DEG. FAHRENHEIT ="; C ;"DEG. CELSIUS"
130 PRINT
140 GOTO 10
```

Se si immette e si esegue questo programma, si vede IN PUT in azione. La riga 10 usa l'istruzione di input non solo per raccogliere informazioni ma anche per stampare la richiesta. Notare inoltre che è possibile chiedere un numero o una stringa (usando uno variabile numerica o stringa).

Le righe 20, 30 e 40 eseguono alcuni controlli su ciò che è stato battuto. Nella riga 20 se non è stato immesso nulla (è stato semplicemente premuto **[RETURN]**) il programma torna alla riga 10 e chiede di nuovo l'input.

Alla riga 30, se è stato battuto F, si sa che l'utente desidera convertire in gradi Celsius una temperatura espressa in gradi Fahrenheit cosicché il programma salta alla parte che esegue quella conversione.

La riga 40 esegue un altro controllo. Sappiamo che ci sono soltanto due scelte valide che l'utente può immettere. Per accedere alla riga 40 l'utente deve battere qualche carattere diverso da F. Ora viene eseguito un controllo per accertarsi se quel carattere è una C; in caso contrario il programma richiede di nuovo l'input.

Tutto ciò può sembrare eccessivamente complicato ma in realtà si tratta di una buona prassi di programmazione.

Un utente che non ha familiarità con il programma può sentirsi molto frustrato se questo fa qualcosa di strano in quanto è stato compiuto un errore nell'immettere le informazioni.

Una volta determinato quale tipo di conversione eseguire, il programma esegue il calcolo e stampa la temperatura immessa e la temperatura convertita.

Il calcolo è semplicemente matematico con l'uso della formula apposita per le conversioni di temperatura. Al termine del calcolo e dopo la stampa del risultato, il programma torna all'inizio e riprende da capo.

Dopo l'esecuzione lo schermo dovrebbe apparire come quello qui di seguito riportato.

```
CONVERT FROM FAHRENHEIT OR CELSIUS (F/C): ? F
ENTER DEGREES FAHRENHEIT: 32
32 DEG. FAHRENHEIT = 0 DEG. CELSIUS

CONVERT FROM FAHRENHEIT OR CELSIUS (F/C): ?
```

Dopo aver eseguito il programma assicurarsi di salvarlo su disco o su nastro. Questo programma nonché gli altri presentati nel corso del manuale possono formare una buona base della propria libreria di programmi.

GET

L'istruzione GET consente di immettere un carattere alla volta da tastiera senza premere **[RETURN]**. Ciò accelera notevolmente l'immissione di dati in molte applicazioni. Qualunque sia il tasto che si preme, viene assegnato alla variabile specificata con GET.

La seguente routine illustra come funziona GET:

```
 1 PRINT "{CLR/HOME}"
10 GET A$: IF A$ = "" THEN 10
20 PRINT A$;         |
30 GOTO 10           +-------------- Qui non c'è spazio
```

Se si esegue (RUN) il programma, lo schermo si cancella e ogni volta che si batte un tasto la riga 20 lo stampa sullo schermo e quindi accede (GET) ad un altro carattere. È importante notare che il carattere immesso non verrà visualizzato a meno che non lo si stampi (PRINT> specificatamente sullo schermo, come è stato fatto in questo caso.

Anche la seconda istruzione sulla riga 10 è molto importante. GET funziona in continuazione anche se non viene premuto alcun tasto (a differenza di INPUT che attende una risposta), cosicché la seconda parte di questa riga controlla in continuazione la tastiera fino a che non viene premuto un tasto.

Vedere cosa succede se la seconda parte della riga 10 viene eliminata.

Per interrompere questo programma è possibile premere i tasti **[RUN/STOP]** e **[RESTORE]**.

La prima parte del programma di conversione della temperatura dovrebbe facilmente essere riscritta per

usare GET. Caricare (LOAD) il programma di conversione della temperatura e modificare le righe 10, 20 e 40 come indicato:

```
10 PRINT "CONVERT FROM FAHRENHEIT OR CELSIUS (F/C) ?"
20 GET A$: IF A$ = "" THEN 20
30 IF A$ <> "C" THEN 20
```

Questa modifica fa sì che il programma funzioni in maniera più uniforme, dato che nulla succede fino a che l'utente non batte una delle risposte desiderate per scegliere il tipo di conversione.

Una volta effettuata questa modifica è bene salvare su nastro o su disco la nuova versione del programma.

NUMERI CASUALI ED ALTRE FUNZIONI

Il COMMODORE 64 contiene numerose funzioni che vengono usate per eseguire operazioni speciali. Le funzioni possono essere viste come programmi incorporati inclusi nel BASIC. Ma anziché battere un certo numero di istruzioni ogni volta che occorre eseguire un calcolo specializzato, basta battere il comando per la funzione desiderata: il computer farà il resto.

Molte volte nel creare un gioco o programma educativo, occorre generare un numero casuale per simulare ad esempio il lancio di dadi. È possibile certamente scrivere un programma che generi questi numeri ma un modo più facile consiste nell'utilizzare la funzione dei numeri RaNDom (casuali).

Per vedere come funziona effettivamente RND, provare con questo breve programma:

```
NEW

10 FOR X = 1 TO 10
20 PRINT RND(1), <--- Se si tralascia la virgola,
30 NEXT               la lista di numeri uscirà come una fila
```

Dopo aver eseguito il programma, lo schermo si presenta come indicato nella figura:

```
.789280697          .664673958
.256373663          .0123442287
.682952381          3.90587279E-04
.402343724          .879300926
.158209063          .245596701
```

I numeri non corrispondono? Se corrispondessero sarebbe un bel problema, in quanto essi devono essere completamente casuali!

Cercare di eseguire il programma altre volte per verificare che i risultati siano sempre diversi. Anche se i numeri non seguono alcun profilo, si può però notare che alcune cose rimangono inalterate ogni volta che il programma viene eseguito.

Innanzitutto i risultati sono sempre compresi tra 0 e i ma mai uguali a 0 o a 1. Ciò non accadrà mai se si vuole simulare il lancio casuale di un dado dato che ci si aspetta numeri compresi tra i e 6.

L'altra caratteristica importante da osservare è che si tratta di numeri reali (con cifra decimale). Ciò potrebbe a sua volta rappresentare un problema dato che occorrono spesso numeri interi.

Esistono numerosi modi semplici per produrre i numeri dalla funzione RND nel campo desiderato.

Sostituire la riga 20 con la seguente ed eseguire di nuovo il programma:

```
20 PRINT 6*RND(1),

RUN

3.60563664          4.53660853
5.47238963          8.40850227
3.19265054          4.39547668
3.16331095          5.50620749
9.32527884          4.17090293
```

Ciò ha risolto il problema di ottenere risultati non superiori a i ma esiste sempre la parte decimale del risultato. A questo punto può essere richiamata un'altra funzione.

La funzione INTeger (intero) converte i numeri reali in valori interi. Ancora una volta sostituire la riga 20 con la seguente ed eseguire il programma per vedere l'effetto della modifica:

```
20 PRINT INT(6*RND(1)),

RUN

2          3          1          0
2          4          5          5
0          1
```

Il programma ha fatto molto per avvicinare all'obiettivo originale di generare numeri casuali compresi tra i e 6. Se si esamina da vicino ciò che è stato generato in questa occasione, si trova che i risultati sono compresi soltanto nel campo da O a 5.

Con l'ultima fase, aggiungere i all'istruzione come segue:

```
20 PRINT INT(6*RND(1))+1,
```

Ora, si è ottenuto il risultato desiderato.

In generale è possibile inserire un numero, una variabile o qualsiasi espressione BASIC tra le parentesi della funzione INT. In relazione al campo desiderato basta moltiplicare il limite superiore per la funzione RND. Ad esempio, per generare numeri casuali tra i e 25 si potrebbe battere;

```
20 PRINI INT(25*RND(1))+1
```

La forma generale per creare una serie di numeri casuali in un certo campo è la seguente:

```
NUMBER = INT (UPPER LIMIT*RND(1)) + LOWER LIMIT
```

GIOCO DEGLI INDOVINELLI

Dato che è stato speso un po' di tempo per conoscere i numeri casuali perché non utilizzare le informazioni apprese?

Il gioco che segue non solo illustra un buon uso dei numeri casuali ma introduce anche ulteriori teorie di programmazione.

Nell'eseguire questo programma verrà generato un numero casuale, NM.

```
NEW

1 REM NUMBER GUESSING GAME
2 PRINT "{CLEAR}"
5 INPUT "ENTER UPPER LIMIT FOR GUESS ";LI
10 NM = INT(LI*RND(1))+1
15 CN = 0
20 PRINT "I'VE GOT THE NUMBER." : PRINT
30 INPUT "WHAT'S YOUR GUESS "; GU
35 CN = CN + 1
40 IF GU > NM THEN PRINT "MY NUMBER IS LOWER." : PRINT : GOTO 30
50 IF GU < NM THEN PRINT "MY NUMBER IS HIGHER.": PRINT : GOTO 30
60 PRINT "GREAT! YOU GOT MY NUMBER"
65 PRINT "IN ONLY"; CN ;"GUESSES." : PRINT
70 PRINT "DO YOU WANT TO TRY ANOTHER (Y/N) ?";
80 GET AN$ : IF AN$ = "" THEN 80
90 IF AN$ = "Y" THEN 2
100 IF AN$ <> "N" THEN 70
```

```
110 END
```

È possibile specificare la dimensione del numero all'inizio del programma. A questo punto occorre indovinare qual è il numero.

Segue ora un'esecuzione esemplificativa unitamente alla spiegazione.

```
ENTER UPPER LIMIT FOR GUESS ? 25
I'VE GOT THE NUMBER.

WHAT'S YOUR NUMBER GUESS ? 15
MY NUMBER IS HIGHER.

WHAT'S YOUR NUMBER GUESS ? 20
MY NUMBER IS LOWER.

WHAT'S YOUR NUMBER GUESS ? 19
GREAT! YOU GOT MY NUMBER
IN ONLY 3 GUESSES.

DO YOU WANT TO TRY ANOTHER (Y/N) ?
```

Le istruzioni IF/THEN confrontano i tentativi di individuare il numero con il numero effettivamente generato. In relazione al numero ipotizzato il programma dice se questo era più alto o più basso del numero casuale generato.

Dalla formula data per determinare il campo di numeri casuali vedere se è possibile aggiungere alcune righe al programma che consenta all'utente di specificare anche il campo inferiore di numeri generati.

ogniqualvolta si fa un'ipotesi, CN è incrementato di 1 per tener nota del numero dei tentativi. Usando il programma, accertarsi se è possibile usare un buon ragionamento per indovinare un numero nel minor numero di tentativi.

Quando si ottiene la risposta esatta, il programma stampa il messaggio «GREAT! YOU GOT MY NUMBER» unitamente al numero di tentativi che sono stati necessari.

È possibile quindi riprendere da capo il procedimento.

Ricordarsi che il programma genera ogni volta un nuovo numero casuale.

SUGGERIMENTI DI PROGRAMMAZIONE:

Nelle righe 40 e 50 viene usato un due punti per separare istruzioni multiple su una sola riga.

Ciò non solo risparmia lavoro di battitura ma nei lunghi programmi consente di risparmiare spazio di memoria.

Notare inoltre nelle istruzioni IF/THEN sulle stesse due righe che è stato istruito il computer a stampare (PRINT) qualche cosa anziché saltare immediatamente a qualche altro punto nel programma.

L'ultimo punto illustra il motivo per l'uso dei numeri di riga a incrementi di 10. Dopo che il programma è stato scritto si è deciso di aggiungere la parte di conteggio. Semplicemente aggiungendo queste nuove righe al termine del programma numerate in modo da farle cadere tra le appropriate righe esistenti, il programma è stato mente modificato.

IL LANCIO DI DADI

Il programma che segue simula il lancio di due dadi. È possibile utilizzarlo così come è oppure usarlo come parte di un gioco più complesso.

```
5 PRINT "CARE TO TRY YOUR LUCK?"
10 PRINT "RED DICE   =";INT(6*RND(1))+1
20 PRINT "WHITE DICE =";INT(6*RND(1))+1
30 PRINT "HIT SPACE BAR FOR ANOTHER ROLL" : PRINT
40 GET A$ : IF A$ = "" THEN 40
50 IF A$ = CHR$(32) THEN 10
```

Si vuole tentare la fortuna?
Da ciò che si è finora imparato sui numeri casuali e sul BASIC, si dovrebbe poter segui re e comprendere ciò che succede.

GRAFICI CASUALI

Come nota finale sui numeri casuali e come introduzione al disegno di grafici, è bene dedicare un istante ad immettere e ad eseguire questo semplice programma:

```
10 PRINT "{CLR/HOME}"
20 PRINT CHR$(205.5+RND(1));
40 GOTO 20
```

Come ci si potrebbe aspettare, la riga chiave è la numero 20. Un'altra funzione, CHR$ (stringa di caratteri) fornisce un carattere basato su un numero di codici standard da O a 255. Ogni carattere che il COMMODORE 64 può stampare è codificato in questo modo (vedere Appendice F).
Per trovare rapidamente il codice di qualsiasi carattere battere:

```
PRINT ASC("X")
```

dove X è il carattere che si sta controllando (può trattarsi di qualsiasi carattere stampabile, compresi i segni grafici). La risposta è il codice per il carattere battuto. Come probabilmente ci si immagina, «ASC» è un'altra funzione che dà il codice standard «ASCII» per il carattere battuto.
È ora possibile stampare quel carattere battendo:

```
PRINT CHR$(X)
```

Se si prova a battere:

```
PRINT CHR$(205); CHR$(206)
```

si vedranno comparire i due caratteri grafici di destra sui tasti M e N. Questi sono i due caratteri che il programma usa per il labirinto.
Usando la formula 205.5 + RND(1) il computer sceglie un numero casuale compreso tra 205.5 e 206.5. C'è una probabilità del 50% che il numero risulti al disopra o al disotto di 206. CHR$ ignora qualsiasi valore frazionario cosicché metà delle volte viene stampato il carattere con il codice 205 mentre le rimanenti volte viene visualizzato il codice 206.
Se si vuole fare qualche esperimento con questo programma, si può ad esempio cercare di cambiare 205.5 aggiungendo o sottraendo un paio di decine. Ciò conferisce ad entrambi i caratteri una maggior possibilità di venir scelti.

CAPITOLO 5

COMANDI AVANZATI PER COLORI E GRAFICI

- Colore e grafici
- Stampa (PRINT) dei colori

- PEEK e POKE
- Grafici sullo schermo
- Altro sulle palline rimbalzanti

COLORE E GRAFICI

Finora sono state esplorate alcune delle sofisticate capacità di calcolo del COMMODORE 64. Ma una delle sue caratteristiche più affascinanti è la sua eccezionale abilità di produrre grafici e colori.
Si è visto un rapido esempio dei grafici nei programmi della pallina "rimbalzante" e del "labirinto". Ma questi hanno soltanto sfiorato la potenza a disposizione. In questo capitolo verranno introdotti numerosi nuovi concetti per spiegare la programmazione dei grafici e dei colori e per mostrare come è possibile creare propri giochi ed esempi di animazione avanzata.
Poiché finora ci si è concentrati sulle capacità di calcolo della macchina, tutti gli schermi generati erano in un solo colore (testo azzurro su fondo blu con un bordo azzurro).
In questo capitolo si vedrà come aggiungere il colore ai programmi e come controllare tutti quegli strani simboli grafici che figurano sulla tastiera.

LA STAMPA DEI COLORI

Come si sarà scoperto nella prova di messa a punto del Capitolo 1, è possibile cambiare i colori del testo semplicemente tenendo abbassato il tasto **[CTRL]** ed uno dei tasti colore. Ciò funziona molto bene nel modo immediato, ma cosa succede se si vuole incorporare cambiamenti di colore nei programmi?
Quando è stato presentato il programma della "pallina rimbalzante" si è visto come è possibile incorporare comandi da tastiera tipo movimento del cursore all'interno delle istruzioni PRINT. In modo analogo è possibile aggiungere modifiche al colore del testo ai programmi.
È possibile lavorare con un'intera serie di 16 colori di testo. Usando il tasto **[CTRL]** ed un tasto numerico, sono disponibili i seguenti colori:

1	2	3	4	5	6	7	8
Nero	Bianco	Rosso	Blu-verde	Porpora	Verde	Blu	Giallo

Se si tiene abbassato il tasto **[C=]** unicamente al tasto di numero appropriato, possono essere usati questi ulteriori otto colori:

1	2	3	4	5	6	7	8
Arancio	Marrone	Rosso Chiaro	Grigio 1	Grigio 2	Verde Chiaro	Azzurro	Grigio 3

Battere NEW e provare con quanto segue. Tenere abbassato il tasto **[CTRL]** e contemporaneamente battere il tasto **[1]**. Successivamente, battere il tasto **[R]** senza però tenere abbassato il tasto **[CTRL]**. Ora premere di nuovo il tasto **[CTRL]** e contemporaneamente premere il tasto **[2]**. Sollevare il tasto **[CTRL]** e premere il tasto **[A]**. Provare con i vari numeri alternandoli con le lettere e battere la parola RAINBOW come segue:

```
10 PRINT"   R    A    I    N    B    O    W"
           |    |    |    |    |    |    |
[CTRL] [1]|[2]|[3]|[4]|[5]|[6]|[7]|

RUN
RAINBOW
```

Nello stesso modo in cui i comandi del cursore compaiono come caratteri grafici all'interno dei segni di virgolette delle istruzioni di stampa, così anche i comandi dei colori sono rappresentati sotto forma di caratteri grafici.
Nel precedente esempio quando si è tenuto abbassato **[CTRL]** e si è battuto **[3]** è comparso il carattere "<-". **[CTRL][7]** hanno fatto comparire "£". Ciascun comando dei colori, usato in questo modo, visualizzerà proprio

codice grafico esclusivo. La tabella mostra le rappresentazioni grafiche di ciascun comando di colore stampabile.

TASTIERA	COLORE SCHERMO	TASTIERA	COLORE SCHERMO
[CTRL][1]	NERO	**[C=][1]**	ARANCIO
[CTRL][2]	BIANCO	**[C=][2]**	MARRONE
[CTRL][3]	ROSSO	**[C=][3]**	ROSSO CHIA.
[CTRL][4]	BLU VERDE	**[C=][4]**	GRIGIO 1
[CTRL][5]	PORPORA	**[C=][5]**	GRIGIO 2
[CTRL][6]	VERDE	**[C=][6]**	VERDA CHIA.
[CTRL][7]	BLU	**[C=][7]**	AZZURRO
[CTRL][8]	GIALLO	**[C=][8]**	GRIGIO 3

Quantunque l'istruzione PRINT possa apparire un poco strana sullo schermo, quando si esegue (RUN) il programma verrà visualizzato soltanto il testo. E questo cambierà automaticamente colore secondo i comandi inseriti nell'istruzione di stampa.

Provare con altri esempi a propria scelta mescolando qualsiasi numero di colori all'interno di un'istruzione PRINT. Ricordarsi anche che è possibile usare la seconda serie di colori di testo servendosi dei tasti COMMODORE e dei tasti numerici.

SUGGERIMENTO:

Si noterà dopo aver eseguito un programma con modifiche di colore o di modo (negativo) che la richiesta "READY" e qualsiasi ulteriore tasto battuto compare nello stesso colore o nello steso modo. Per ritornare alla presentazione normale, ricordarsi di premere **[RUN/STOP]** e **[RESTORE]**.

CODICI CHR$ DEI COLORI

Prima di procedere con questo capitolo, occorre dare un breve sguardo all'Appendice F.

Si è notato nell'osservare la lista dei codici CHR$ nell'Appendice F che ciascun colore (nonché la maggior parte degli altri comandi di tastiera, ad esempio i movimenti del cursore) hanno un codice esclusivo. Questi codici possono essere stampati direttamente per ottenere gli stessi risultati della pressione del tasto **[CTRL]** è dell'appropriato tasto all'interno dell'istruzione PRINT.

Provare con questo esempio:

```
NEW
10 PRINT CHR$(147) : REM {CLR/HOME)
10 PRINT CHR$(30);"CHR$(30) CHANGES ME TO?"
RUN
CHR$(30) CHANGES ME TO?
```

Il testo dovrebbe ora risultare verde. In molti casi l'uso della funzione CHR$ sarà molto facile, particolarmente se si vuole provare a cambiare i colori. Alla pagina seguente c'è un modo diverso per ottenere un arcobaleno di colori. Dato che c'è un numero di righe simili (40-110) usare i tasti di editing per risparmiare molta battitura. Vedere le note dopo la lista per rinfrescare la memoria sulle procedure dì correzione.

```
NEW

1 REM AUTOMATIC COLOR BARS
5 PRINT CHR$(147) : REM CHR$(147)= CLR/HOME
10 PRINT CHR$(18); "    " ;:REM REVERSE BAR
20 CL = INT(8*RND(1))+1
30 ON CL GOTO 40,50,60,70,80,90,100,110
40 PRINT CHR$(5);: GOTO 10
50 PRINT CHR$(28);: GOTO 10
60 PRINT CHR$(30);: GOTO 10
70 PRINT CHR$(31);: GOTO 10
80 PRINT CHR$(144);: GOTO 10
90 PRINT CHR$(156);: GOTO 10
```

```
100 PRINT CHR$(158);: GOTO 10
110 PRINT CHR$(159);: GOTO 10
```

Battere normalmente le righe da 5 a 40. Lo schermo dovrebbe apparire
come segue:

```
1 REM AUTOMATIC COLOR BARS
5 PRINT CHR$(147) : REM CHR$(147)= CLR/HOME
10 PRINT CHR$(18); "     ";:REM REVERSE BARS
20 CL = INT(8*RND(1))+1
30 ON CL GOTO 40,50,60,70,80,90,100,110
40 PRINT CHR$(5);: GOTO 10
```

NOTE DI CORREZIONE

Usare il tasto di movimento del cursore verso l'alto per posizionare il cursore alla riga 40. Quindi battere 5 sopra il 4 di 40. Successivamente usare il tasto per lo spostamento a destra del cursore per portarsi sul 5 nelle parentesi CHR$. Premere **[SHIFT] [INST/DEL]** per creare uno spazio e battere «28». Ora premere semplicemente **[RETURN]** con il cursore disposto ovunque sulla riga.

Lo schermo dovrebbe presentarsi come segue:

```
1 REM AUTOMATIC COLOR BARS
5 PRINT CHR$(147) : REM CHR$(147)= CLR/HOME
10 PRINT CHR$(18); "      " ;:REM REVERSE BAR
20 CL = INT(8*RND(1))+1
30 ON CL GOTO 40,50,60,70,80,90,100,110
50 PRINT CHR$(28);: GOTO 10
```

Non è il caso di preoccuparsi. La riga 40 c'è ancora. Basta listare (LIST) il programma e vedere. Usando la stessa procedura continuare a modificare l'ultima riga con un nuovo numero di riga ed il codice CHR$ tino a che non sono state immesse tutte le righe rimanenti.

Come controllo finale, listare l'intero programma prima di eseguirlo per assicurarsi che tutte le righe siano state immesse correttamente(RUN)

Ecco una breve spiegazione di quello che sta succedendo.

È stata finora probabilmente rappresentata la maggior parte del programma delle barre a colori salvo qualche nuova strana istruzione alla riga 30. Ma si può rapidamente osservare cosa in effetti fa l'intero programma.

La riga 5 stampa il codice CHR$ per CLR/HOME**.**

La riga 10 attiva le scritte in negativo e stampa 5 spazi, che si trasformano in una barra dato che sono in negativo. La prima volta nel corso del programma la barra sarà azzurra, il normale colore del testo.

La riga 20 usa la funzione del numero casuale per scegliere un colore a caso tra 1 e 8.

La riga 30 continua una variazione dell'istruzione IF...THEN che è detta ON...GOTO e che consente al programma di scegliere da una lista di numeri ai quali andare. Se la variabile (in questo CL) ha un valore di 1, il primo numero di riga è quello scelto (in questo caso 40). Se il valore è 2, viene usato il secondo numero della lista, ecc.

Le righe da 40 a 110 convertono semplicemente i colori casuali dei tasti nell'appropriato codice CHR$ per quel colore e riportano il programma alla riga 1O per stampare (PRINT) una sezione della barra in quel colore. Quindi l'intero processo riparte da capo.

Provare a produrre 16 numeri casuali, a espandere ON...GOTO per manipolarli e ad aggiungere i rimanenti codici CHR$ per visualizzare i rimanenti otto colori.

PEEK e POKE

Non ci proponiamo ora di sezionare il computer ma semplicemente di dare uno sguardo all'interno della macchina per «inserire» determinate cose in qualche punto.

cose in qualche punto.

Esattamente come le variabili possono essere viste come una rappresentazione di «scatola» all'interno della macchina nelle quali si inseriscono informazioni, cosi è possibile pensare che alcune «scatole» particolarmente

definite all'interno del computer rappresentino locazioni
specifiche di memoria.

Il COMMODORE 64 esamina queste locazioni di memoria per rendersi conto di quale deve essere il colore dello sfondo e del bordo, quali caratteri devono essere visualizzati sullo schermo - e dove - e per numerosi altri motivi.

Inserendo («POKE») un valore diverso nell'appropriata locazione di memoria è possibile cambiare colore, definire e spostare oggetti ed addirittura creare musica.

Queste locazioni di memoria possono essere rappresentate come segue:

```
     53280              53281              53282              53283
       X                  Y

     COLORE          COLORE DELLO
   DEL BORDO            SFONDO
```

A pagina 60 sono state indicate soltanto quattro locazioni, due delle quali controllano i colori dello sfondo e dello schermo. Provare a battere istruzione che segue:

POKE 53281,7 (e poi battere RETURN)

Il colore dello sfondo dello schermo cambierà in giallo in quanto è stato inserito Il valore 7 -corrispondente al giallo - nella locazione che controlla il colore dello sfondo.

Provare ad inserire (POKE) diversi valori nella locazione del colore dello sfondo ed osservare i relativi risultati. È possibile inserire (POKE) qualsiasi valore compreso fra O e 255 ma di essi soltanto quelli da O a 15 funzioneranno.

I valori effettivi per inserire (POKE) ciascun colore sono:

0	NERO	8	ARANCIO
1	BIANCO	9	MARRONE
2	ROSSO	10	ROSSO CHIARO
3	CELESTE	11	GRIGIO 1
4	PORPORA	12	GRIGIO 2
5	VERDE	13	VERDE CHIARO
6	BLU	14	AZZURRO
7	GIALLO	15	GRIGIO 3

Si può pensare ad un modo per visualizzare le varie combinazioni di colore dello sfondo e del bordo? Il programma che segue può essere di qualche aiuto:

```
NEW

10 FOR BA = 0 TO 15
20 FOR BO = 0 TO 15
30 POKE 53280, BA
40 POKE 53281, BO
50 FOR X = 1 TO 2000: NEXT X
60 NEXT BO: NEXT BA

RUN
```

Sono stati in questo caso create due semplici iterazioni per inserire (POKE) i vari valori in modo da cambiare i colori del fondo e del bordo. L'iterazione DELAY (ritardo) alla riga 50 serve soltanto a rallentare il movimento.

Chi è curioso può provare anche il programma che segue:

?PEEK (53280) AND 15

Si dovrebbe ottenere un valore di 15. Questo è l'ultimo valore attribuito al bordo ed ha un senso in quanto sia il colore dello sfondo che del bordo è il grigio (valore 15) dopo che il programma è eseguito.

Immettendo AND 15 si eliminano tutti gli altri valori salvo 1-15 a causa del modo in cui i codici dei colori sono memorizzati nel computer. Normalmente ci si aspetterà di trovare lo stesso valore che è stato inserito (POKE) per ultimo nella locazione.

In generale PEEK consente di esaminare una locazione specifica e vedere quale valore è in essa presente. E' possibile pensare all'aggiunta di una riga al programma che visualizza il valore dello sfondo e del bordo mentre il programma viene eseguito? Eccola:

```
25 PRINT CHR$(147); "BORDER = ";PEEK(53280) AND 15, "BACKGROUND = ";PEEK(53281) AND
15
```

GRAFICI SULLO SCHERMO

In tutta la stampa delle informazioni che è stata finora eseguita, il computer ha trattato le informazioni in modo sequenziale, stampando cioè un carattere dopo l'altro, iniziando dalla posizione corrente del cursore (salvo nel caso in cui si è chiesta una nuova riga o è stato usato «,» nella formattazione di PRINT).

Per stampare (PRINT) i dati in un particolare punto è possibile iniziare da un punto noto sullo schermo e PRINT il numero appropriato di comandi del cursore per formattare lo schermo. Ma ciò richiede passi di programma e molto tempo.

Esattamente come ci sono taluni punti nella memoria del COMMODORE 64 per controllare i colori così ce ne sono anche alcuni che è possibile usare per controllare direttamente ciascuna locazione sullo schermo.

LA MAPPA DI MEMORIA DELLO SCHERMO

Dato che lo schermo del computer è in grado di contenere 1000 caratteri (40 colonne per 25 righe), significa che si sono 1000 locazioni di memoria accantonate per gestire ciò che viene posto sullo schermo stesso. La struttura dello schermo può essere vista come una griglia, in cui ciascun quadrato rappresenta una locazione di memoria.

È dato che ciascuna locazione della memoria può contenere un numero da O a 255, ci sono 256 possibili valori per ciascuna locazione di memoria. Questi valori rappresentano i diversi caratteri che COMMODORE 64 può visualizzare (vedere Appendice E).

Inserendo (POKE) il valore di un carattere nell'appropriata locazione di memoria dello schermo, quel carattere viene visualizzato nella posizione corrispondente.

[IMMAGINE PAG 63]

La memoria dello schermo nel COMMODORE 64 inizia normalmente alla locazione di memoria 1024 e termina alla locazione 2023. la locazione 1024 si trova nell'angolo superiore sinistro dello schermo. La locazione 1025 è la posizione del successivo carattere alla sua destra e così via lungo la fila. La locazione 1063 è la posizione più a destra della prima fila. La successiva locazione che segue l'ultimo carattere sulla fila è il primo carattere della fila successiva.

Si supponga di voler controllare una pallina che rimbalza sullo schermo.

La pallina si trova nel mezzo dello schermo, sulla colonna 20 fila 12. La formula per il calcolo della locazione di memoria sullo schermo è la seguente:

```
POINT = 1024 + X +40*Y
```

Dove X è la colonna e Y è la fila.

Pertanto la locazione di memoria della pallina è:

```
1024 + 20 + 480 oppure 1524
       |    |
       |    +---- FILA (40*12)
       |
       +--------- COLONNA
```

Cancellare ora lo schermo con SHIFT e CLR/HOME e battere:

```
POKE 1524,81 <--- CODICE DEL CARATTERE
       |
       +--- LOCAZIONE
```

MAPPA DELLA MEMORIA DEI COLORI

Compare una pallina al centro dello schermo? Ciò significa che si è inserito un carattere direttamente nella memoria dello schermo senza usare l'istruzione PRINT.

La pallina comparsa era bianca. C'è comunque un modo per cambiare il colore di un oggetto sullo schermo alterando un altro campo di memoria.

Battere:

```
POKE 55796,2 <--- COLORE
       |
       +--- LOCAZIONE
```

Il colore della pallina cambia in rosso. Per ogni punto sullo schermo del COMMODORE 64 ci sono due locazioni di memoria, una per il codice del carattere e l'altra per il Codice del colore. La mappa di memoria dei colori inizia alla locazione 55296 (angolo superiore sinistro) e Continua per 1000 locazioni.

[IMMAGINE PAG 65]

Gli stessi codici dei colori da O a 15 che sono stati usati per cambiare i colori del bordo e dello sfondo qui possono essere usati per cambiare direttamente i colori dei caratteri.

La formula usata per calcolare le locazioni di memoria dello schermo può essere modificata per fornire le locazioni in cui inserire (POKE) i codici dei colori.

La nuova formula è:

```
COLOR PRINT = 55296 + X + 40*Y
```

ALTRO SULLE PALLINE RIMBALZANTI

Qui c'è un programma revisionato per la pallina rimbalzante che stampa direttamente sullo schermo con i POKE anziché usando i comandi del cursore all'interno delle istruzioni PRINT. Come si vedrà dopo aver eseguito il programma, la flessibilità è maggiore che non quella precedente e consente di programmare un'animazione molto più sofisticata.

```
NEW
10 PRINT "{CLEAR}"
20 POKE 53280,7 : POKE 53281,13
30 X = 1 : Y = 1
40 DX = 1 : DY = 1
50 POKE 1024 + X + 40*Y,81
60 FOR T = 1 TO 10 : NEXT
70 POKE 1024 + X + 40*Y,32
80 X = X + DX
90 IF X = 0 OR X = 39 THEN DX = -DX
100 Y = Y + DY
110 IF Y = 0 OR Y = 24 THEN DY = -DY
120 GOTO 50
```

La riga 10 cancella lo schermo e la riga 20 definisce lo sfondo al verde chiaro con un bordo giallo.

Le variabili X e Y nella riga 30 tengono nota della posizione corrente di fila e di colonna della pallina. Le variabili DX e DY nella riga 40 rappresentano la direzione orizzontale e verticale del movimento della pallina. Quando viene aggiunto +1 al valore X, la pallina viene spostata verso destra; quando viene aggiunto -1 la pallina viene spostata verso sinistra. L'aggiunta di +1 a Y sposta la pallina verso il basso di una fila; l'aggiunta di -1 a Y sposta la pallina verso l'alto di una fila.

La riga 50 inserisce la pallina sullo schermo nella posizione corrente del cursore. La riga 60 è l'ormai nota iterazione di ritardo che lascia la pallina sullo schermo quanto basta per poterla osservare.

La riga 70 cancella la pallina inserendo uno spazio (codice 32) nel punto in cui essa si trovava precedentemente sullo schermo.

La riga 80 raggiunge il fattore di direzione a X. La riga 90 si accerta che la pallina abbia raggiunto una delle pareti laterali, invertendo la direzione se c'è un rimbalzo. La riga 100 e 110 esegue la stessa cosa per le pareti superiore ed inferiore.

La riga 120 riporta il programma allo schermo e sposta di nuova la pallina.

Cambiando il codice nella riga 50 da 81 ad un altro codice di carattere, è possibile trasformare la pallina in qualsiasi altro carattere. Se si cambia DX o DY in zero, la pallina rimbalza in maniera diritta anziché in diagonale.

È inoltre possibile aggiungere qualche altra informazione.

Finora le sole cose controllate erano i valori X e Y che uscivano dai limiti dello schermo. Aggiungere le seguenti righe al programma.

```
21 FOR L = 1 TO 10
25 POKE 1024 + INT(RND(1)*1000), 102
27 NEXT L
115 IF PEEK(1024 + X + 40*Y) = 102 THEN DX = -DX:GOTO 80
```

Le righe da 21 a 27 inseriscono 10 blocchi sullo schermo in posizioni casuali. La riga 115 controlla (PEEK) che la pallina sia sul punto da rimbalzare in un blocco ed in questo caso ne cambia la direzione.

```
10 PRINT "{CLEAR}"
20 POKE 53280,7 : POKE 53281,13
21 FOR L = 1 TO 10
25 POKE 1024 + INT(RND(1)*1000), 102
27 NEXT L
30 X = 1 : Y = 1
40 DX = 1 : DY = 1
50 POKE 1024 + X + 40*Y,81
60 FOR T = 1 TO 10 : NEXT
70 POKE 1024 + X + 40*Y,32
80 X = X + DX
90 IF X = 0 OR X = 39 THEN DX = -DX
100 Y = Y + DY
110 IF Y = 0 OR Y = 24 THEN DY = -DY
115 IF PEEK(1024 + X + 40*Y) = 102 THEN DX = -DX:GOTO 80
120 GOTO 50
```

CAPITOLO 6

GRAFICI ANIMATI (SPRITES)

- Introduzione ai grafici animanti
- Creazione di effetti di animazione
- Ulteriori note sugli effetti di animazione
- Aritmetica binaria

INTRODUZIONE AI GRAFICI ANIMATI

Nei precedenti capitoli che si occupavano dei grafici si è visto che i simboli grafici possono essere usati nelle istruzioni PRINT per creare effetti di animazione ed aggiungere altre caratteristiche agli schermi.

E' stato inoltre presentato un modo di inserire (POKE) codici di caratteri in specifiche locazioni di memoria dello schermo, in modo da far comparire direttamente gli appropriati caratteri nel punto voluto sullo schermo.

La creazione di effetti di animazione in entrambi questi casi richiede una massa notevole di lavoro in quanto si devono creare gli oggetti dagli esistenti simboli grafici. Lo spostamento dell'oggetto richiede un certo numero di istruzioni di programma per seguire l'oggetto e portarlo in una nuova posizione. E a causa delle limitazioni nell'uso dei simboli grafici, il profilo e la risoluzione dell'oggetto potrebbero non essere quelli ideali.

Usando i grafici animati si elimina una buona parte di questi problemi. Con il termine "grafico animato" s'intende un oggetto programmabile ad alta risoluzione che può essere creato attribuendogli pressoché qualsiasi profilo - attraverso i comandi BASIC. L'oggetto può essere facilmente spostato sullo schermo semplicemente indicando al computer la posizione che l'oggetto deve occupare. Il computer si occupa di tutto il resto.

Ma i grafici animati hanno anche altre capacità: il loro colore può essere cambiato, è possibile dire se un oggetto entra in collisione con un altro, li si può disporre in modo che uno si porti davanti o dietro un altro e le loro dimensioni possono venir facilmente ampliate.

L'impegno per far tutto ciò è minimo ma per poter ottenere effetti di animazione occorre conoscere qualche altro dettaglio sul modo in cui il COMMODORE 64 opera e sul modo in cui i numeri sono manipolati all'interno del computer. Non è tuttavia difficile quanto sembra: basta seguire gli esempi per poter creare propri effetti di animazione o far compiere agli oggetti le cose più strane.

CREAZIONE DI EFFETTI ANIMAZIONE

Gli effetti di animazione sono controllati da uno speciale dispositivo per la creazione di immagini incorporato nel COMMODORE 64. Questo dispositivo gestisce lo schermo video svolgendo tutto il lavoro di creare e di seguire i caratteri ed i grafici, creando colori e spostandoli sullo schermo.

Il circuito dello schermo dispone di 46 diverse locazioni "ON/OFF" che agiscono come locazioni interne di memoria. Ciascuna di queste locazioni si suddivide in una serie di otto blocchi. E ciascun blocco può essere a sua volta "on" o "off" (rispettivamente "attivato" o "disattivato"). Ma di ciò si parlerà più a lungo in seguito. Inserendo (POKE) l'appropriato valore decimale nella corretta locazione di memoria e possibile controllare gli effetti di animazione.

Oltre ad accedere a molte delle locazioni che creano immagini si userà anche una parte della memoria principale di COMMODORE 64 per memorizzare le informazioni (dati) che definiscono i grafici animati. Infine, verranno usate otto locazioni di memoria immediatamente dopo la memoria dello schermo per dire al computer esattamente da quale area di memoria ciascun disegno animato ricaverà i suoi dati.

Man mano si procederà con gli esempi il processo apparirà abbastanza lineare e se ne comprenderà agevolmente il funzionamento.

Procediamo quindi con la creazione di alcuni disegni animati. Un oggetto animato può misurare 24 punti di larghezza per 21 di lunghezza e si possono controllare fino a 8 disegni animati alla volta. Le animazioni sono visualizzate in un modo speciale ad alta risoluzione che trasforma lo schermo in un'area larga 320 punti ed alta 200 punti.

Si voglia ad esempio creare un pallone e farlo svolazzare nel cielo. Il pallone può essere disegnato nella griglia 24 x 21 (pagina 70).

La fase successiva consiste nel convertire il disegno grafico in dati che il computer può usare. Occorre a questo punto procurarsi un blocco per appunti o carta per grafici e disegnare una semplice griglia che misuri 21 spazi in altezza e 24 in larghezza. Nella parte superiore scrivere 128, 64, 32, 16, 8, 4, 2, 1 per tre volte (come indicato) per ciascuno dei quadrati larghezza 24. Numerare il lato sinistro della griglia da 1 e 21. Scrivere la parola DATA al termine di ciascuna riga. Riempire ora la griglia con un disegno a piacere oppure usare il pallone riportato in figura. È più facile disegnare per prima cosa il profilo esterno e quindi lavorare all'interno riempiendo la griglia.

Ora considerando "on" tutti i quadratini riempiti, sostituire un 1 a ciascuno di essi. I quadrati vuoti sono quelli considerati "off" e ad essi corrisponde uno zero.

Iniziando sulla prima riga, occorre convertire i punti in tre elementi di dati separati che il computer può leggere. Ciascuna serie di otto quadrati nel pallone corrisponde ad un elemento di dati detto byte. Procedendo da sinistra, i primi otto quadrati sono vuoti, ossia 0, così il valore per quella serie di numeri è 0.

```
          SERIE  |SERIE  |SERIE
             1   |   2   |   3
                 |       |
          1       1       1
          2631    2631    2631
          842684218426842184268421
         +------------------------+
       1 |.........#######........|
       2 |.......###########......|
       3 |......#############.....|
       4 |......#####...#####.....|
       5 |.....#####.###..####....|
       6 |.....#####.###.#####....|
       7 |.....#####.###..####....|
       8 |......#####...#####.....|
       9 |......#############.....|
  R   10 |......#############.....|
  I   11 |......#.#########.#.....|
  G   12 |.......#.#######.#......|
  A   13 |.......#..#####..#......|
      14 |........#..###..#.......|
      15 |........#..###..#.......|
      16 |.........#..#..#........|
      17 |.........#..#..#........|
      18 |..........#####.........|
      19 |..........#####.........|
      20 |..........#####.........|
      21 |...........###..........|
         +------------------------+
                   1    1    2   2
          1   5    0    5    0   4

                COLONNA
```

La serie intermedia assomiglia a quella che segue (di nuovo 1 indica un quadratino pieno, 0 un quadratino vuoto):

```
 128    64    32    16    8     4     2     1
           +-----+-----+-----+-----+-----+-----+-----+-----+
           |  0  |  1  |  1  |  1  |  1  |  1  |  1  |  1  |
           +-----+-----+-----+-----+-----+-----+-----+-----+
              ^     ^     ^     ^     ^     ^     ^     ^
              |     |     |     |     |     |     |     |
              0  +  64 +  32 +  16 +  8  +  4  +  2  +  1  =  127
```

La terza serie sulla prima riga contiene a sua volta degli spazi vuoti cosicché anch'essa è uguale a zero. Pertanto i dati per la prima riga sono:

```
DATA 0,127,0
```

Le serie che costituiscono la riga due sono calcolate come segue:

```
           +-----+-----+-----+-----+-----+-----+-----+-----+
  Serie  1:|  0  |  0  |  0  |  0  |  0  |  0  |  0  |  1  |
           +-----+-----+-----+-----+-----+-----+-----+-----+
                                                      1  =  1
```

```
            +-----+-----+-----+-----+-----+-----+-----+-----+
  Serie  2: |  1  |  1  |  1  |  1  |  1  |  1  |  1  |  1  |
            +-----+-----+-----+-----+-----+-----+-----+-----+
               ^     ^     ^     ^     ^     ^     ^     ^
               |     |     |     |     |     |     |     |
              128 +  64 +  32 +  16 +  8  +  4  +  2  +  1  =  255

            +-----+-----+-----+-----+-----+-----+-----+-----+
  Serie  3: |  1  |  1  |  0  |  0  |  0  |  0  |  0  |  0  |
            +-----+-----+-----+-----+-----+-----+-----+-----+
               ^     ^
               |     |
              128 +  64                                     =  192
```

Per la riga 2, i dati sarebbero:

```
DATA 1, 255, 192
```

Allo stesso modo le tre serie che costituiscono ciascuna riga rimanente vengono convertite nel rispettivo valore decimale. Procedere al resto della conversione di questo esempio.

Ora che si dispone dei dati per l'oggetto, come è possibile usarli?

Battere il seguente programma e vedere cosa succede:

```
  1 REM UP, UP, AND AWAY!
  5 PRINT "{CLR/HOME}"
 10 V = 53248  : REM START OF DISPLAY CHIP
 11 POKE V+21, 4  : REM ENABLE SPRITE 2
 12 POKE 2042, 13  : REM SPRITE 2 DATA FROM 13TH BLOCK
 20 FOR N = 0 TO 62  : READ Q : POKE 832+N, Q : NEXT
 30 FOR X = 0 TO 200        ^------ RICAVA LE INFORMAZIONI DA DATA*
 40 POKE V+4, X  : REM UPDATE X COORDINATES
 50 POKE V+5, X  : REM UPDATE Y COORDINATES
 60 NEXT X
 70 GOTO 30
200 DATA 0,127,0,1,255,192,3,255,224,3,231,224 <- INFO. LETTE DA "Q"
210 DATA 7,217,240,7,223,240,7,217,240,3,231,224
220 DATA 3,255,224,3,255,224,2,255,160,1,127,64
230 DATA 1,62,64,0,156,128,0,156,128,0,73,0,0,73,0
240 DATA 0,62,0,0,62,0,0,62,0,0,28,0
```

Se tutto è stato battuto correttamente, il pallone veleggia dolcemente nel cielo (pagina 72).

Per comprendere cosa è successo, occorre per prima cosa sapere quali sono le locazioni che creano le immagini che controllano le funzioni necessarie allo scopo. Queste locazioni, dette registri, possono essere illustrate in questo modo:

Registro (i)	*Descrizione*
0	Coordinata X del disegno animato 0
1	Coordinata X del disegno animato 0
2-15	Abbinato come 0 e 1 per i disegni animati 1-7
16	Bit più significativo - Coordinata X
21	Compra il disegno animato: 1 = appare, 0 = scompare
29	Espande il disegno animato in direzione "X"
23	Espande il disegno animato in direzione "Y"
39-46	Disegno animato colore 0- 7

[IMMAGINE PAG 72]

Fotografia effettiva dello schermo

Oltre a queste informazioni occorre sapere da quale sezione di 64 blocchi ciascuna serie di 8 blocchi di memoria otterrà i disegni animati nei rispettivi dati (1 serie non è stata usata).

Questi dati sono gestiti dalle 8 locazioni che seguono la memoria dello schermo:

```
    +------+------+------+------+------+------+------+------+
    | 2040 | 2041 | 2042 | 2043 | 2044 | 2045 | 2046 | 2047 |
    +------+------+------+------+------+------+------+------+
        ^      ^      ^      ^      ^      ^      ^      ^
        |      |      |      |      |      |      |      |
 SPRITE 0      1      2      3      4      5      6      7
```

Verrà ora descritta la procedura esatta per far muovere i disegni ed infine verrà scritto il programma relativo.

Sono poche le cose necessarie per creare e spostare un oggetto.

1. Far comparire il disegno o i disegni appropriati sullo schermo inserendoli (POKE) nella locazione 21 che lo attiva.

2. Definire il puntatore del disegno (locazioni 2040 - 7) nel punto in cui devono essere letti i dati del disegno.

3. Inserire (POKE) i dati effettivi in memoria.

4. Attraverso un'iterazione, aggiornare le coordinate X e Y per far muovere il disegno.

5. Facoltativamente, è possibile espandere l'oggetto, cambiarne i colori oppure eseguire numerose funzioni speciali. Usare la locazione 29 per espandere il disegno nella direzione "X" e la locazione 23 per espanderlo nella direzione "Y".

Nel programma ci sono soltanto poche cose che potrebbero non essere note dai punti finora discussi.

Nella riga 10;
V = 53248

definisce V alla locazione di partenza della memoria del chip del video. In questo modo si aumenta semplicemente V del numero necessario per ottenere la locazione effettiva di memoria. I numeri di registro sono quelli indicati nella mappa dei registri.

Nella riga 11,
POKE V+21,4

fa sì che compaia il disegno 2 inserendo un 4 in quello che viene chiamato registro di abilitazione (21) per attivare il disegno 2. Si può pensare a ciò nel modo seguente:

```
                          DISEGNI

            Valori decimali per ciascun numero di disegno
        128     64      32      16      8       4       2       1

                    Numero di livello di disegno
         7       6       5       4       3       2       1       0
     +-------+-------+-------+-------+-------+-------+-------+-------+
  21 |   0   |   0   |   0   |   0   |   0   |   1   |   0   |   0   | = 4
     +-------+-------+-------+-------+-------+-------+-------+-------+
                                                 |
                                                 |
                                   Inserire un 1 per il disegno desiderato
```

Ciascun livello di disegno è rappresentato nella sezione 21 della memoria dei disegni e 4 risulta essere il

livello di disegno 2. Se si usasse il livello 3 occorrerebbe inserire un 1 nel disegno 3 che ha un valore di 8. In effetti se si usassero entrambi i disegni 2 e 3 si inserirebbe un 1 sia in 4 che in 8. Occorrerebbe quindi sommare i numeri esattamente come si e fatto con i DATA sulla carta per grafici. Cosi l'attivazione dei disegni 2 e 3 verrebbe rappresentata come V +21,12.

Nella riga 12;
POKE 2042,13

istruisce il computer ad ottenere i dati per il disegno 2 (locazione 2042) dalla tredicesima area della memoria. Si sa dalla costruzione del disegno che esso occupa 63 sezioni di memoria. La cosa potrebbe essere passata inosservata, ma i numeri che sono stati posti nella parte superiore della griglia corrispondono a 3 byte del computer. In altre parole ciascuna serie dei seguenti numeri - 128, 64, 32, 16, 8, 4, 2, 1 equivale ad 1 byte di memoria del computer.

Pertanto con le 21 file della griglia moltiplicate per 3 byte per ciascuna fila, ciascun disegno occupa 63 byte di memoria.

20 FOR N -0 TO 62: READ Q: POKE 832+N,Q: NEXT

Questa riga si occupa della creazione effettiva del disegno. I 63 byte di dati che rappresentano il disegno creato vengono letti (READ) nell'iterazione ed inseriti (POKE) nel tredicesimo blocco di memoria. Questo inizia alia locazione 832.

```
30 FOR X - 0 TO 200
40 POKE V + 4,X
            |
            +----------COORDINATA X DEL DISEGNO 2

50 POKE V + 5,X
            |
            +----------COORDINATA Y DEL DISEGNO 2
```

Si ricorderà che la coordinata X rappresenta il movimento orizzontale di un oggetto sullo schermo e la coordinata Y ne rappresenta il movimento verticale. Pertanto quando i valori di X cambiano nella riga 30 da 0 a 200 (un numero alla volta), il disegno si muove sullo schermo verso il basso e verso destra di uno spazio per ogni numero. I numeri vengono letti (READ) dal computer abbastanza velocemente per far sì che il movimento appaia continuo e non a scatti. Se occorrono ulteriori dettagli basta dare uno sguardo alla mappa dei registri nell'Appendice 0.

Volendo muovere più oggetti, sarebbe impossibile per una sezione di memoria aggiornare contemporaneamente le locazioni di tutti e otto gli oggetti. Pertanto ciascun disegno ha una propria serie di due sezioni di memoria che gli consentono di muoversi sullo schermo.

La riga 70 inizia il ciclo da capo dopo una passata sullo schermo. Il resto del programma e costituito dai dati per il pallone. Naturalmente appaiono ben diversi sullo schermo, non e vero?

Provare ora ad aggiungere la riga seguente:

```
25 POKE V+23,4: POKE V+29,4: REM EXPAND
```

ed eseguire (RUN) il programma di nuovo. II pallone si è allargato raddoppiando la sua misura originale e tutto è avvenuto in modo molto semplice. Inserendo (POKE) 4 (di nuovo per indicare il disegno 2) nelle sezioni di memoria 23 e 29, il disegno è stato ampliato in direzione X e Y.

È importante notare che il movimento inizierà nell'angolo superiore sinistro dell'oggetto. Espandendo l'oggetto in entrambe le direzioni il punto di partenza rimane lo stesso. Per ulteriore divertimento, apportare le seguenti modifiche:

```
11 POKE V+21,12
12 POKE 2042,13: POKE 2043,13
30 FOR X = 1 TO 190
45 POKE V+6,X
55 POKE V+7,190-X
```

È stato creato un secondo disegno (numero 3) inserendo (POKE) 12 nella locazione di memoria che fa sì che compaia il disegno (V+21). Il 12 attiva i disegni 3 e 2 (00001100 = 12).

Le righe aggiunte 45 e 55 spostano il disegno 3 inserendo (POKE) valori nelle 10Cazioni delle Coordinate X e y del disegno (V+6 e V+7).

Si vuole riempire il cielo con ulteriore azione? Basta apportare queste aggiunte:

```
                                                 28 E' IN EFFETTI 4 (DISEGNO 2)
11 POKE V+21 ,28      <---------------- 8 (DISEGNO 3) + 16 (DISEGNO 4)
12 POKE 2042,13: POKE 2043,13: POKE 2044,13
25 POKE V+23,12: POKE V+29,12
48 POKE V+8,X
58 POKE V+9,100
```

Nella riga 11 stavolta è stato fatto in modo di far apparire un altro disegno (4) inserendo (POKE) 28 nell'appropriata locazione "on" della sezione di memoria del disegno. Ora sono attivati i disegni 2-4 (00011100 = 28).

La riga 12 indica che il disegno 4 ricaverà i suoi dati dalla stessa area di memoria (tredicesima area da 63 sezioni) come gli altri disegni inserendo (POKE) 2044,13.

Nella riga 25 i disegni 2 e 3 sono ampliati inserendo (POKE) 12 (disegno 2 e 3 attivati) nelle locazioni di memoria ampliate in direzione X e Y (V+23 e V+29).

La riga 48 sposta il disegno 3 lungo l'asse X. La riga 58 posiziona il disegno 3 a metà strada sullo schermo alla locazione 100. Poiché questo valore non cambia, come aveva fatto in precedenza con X da 0 a 200, il disegno 3 si muove soltanto orizzontalmente.

NOTE ULTERIORI SUI DISEGNI ANIMATI

Dopo tutto questo divertimento con i disegni e le loro possibilità di animazione, sono necessarie alcune altre spiegazioni. Innanzitutto è possibile cambiare il colore del disegno in uno qualsiasi dei 16 colori standard (da 0 a 15) che sono stati usati per cambiare il colore dei caratteri. Vedere il Capitolo 5 oppure l'Appendice G.

Per esempio, per cambiare in verde chiaro il disegno 1. battere: POKE V+40, 13 (assicurarsi di porre V=53248).

Si sarà notato, usando i programmi esemplificativi, che l'oggetto non si è mai spostato verso il margine destro dello schermo e ciò in quanto lo schermo è largo 320 punti e il registrato della direzione X può soltanto contenere un valore di 255. Come può dunque un oggetto spostarsi sull'intero schermo?

Nella mappa di memoria c'è una locazione che non e stata finora citata. Si tratta della locazione 16 (della mappa) che controlla qualcosa chiamato "bit più significativo" (MSB) della locazione di direzione X del disegno. In effetti ciò consente di spostare il disegno in un punto orizzontale compreso tra 256 e 320.

Il MSB del registro X funziona in questo modo: dopo che il disegno e stato spostato alla locazione X 255, inserire un valore nella locazione di memoria 16 che rappresenta il disegno che si vuole spostare. Per esempio per far spostare il disegno 2 alle locazioni orizzontali 256-320, inserire (POKE) il valore del disegno 2 che e (4) nella locazione di memoria 16:

POKE V+16,4.

Iniziare ora a spostare nella direzione abituale X il registro per il disegno 2 (che e nella locazione 4 della mappa) iniziando di nuovo da 1. Dato che ci si sta spostando sol tanto di altri 64 spazi, le locazioni X dovrebbero stavolta essere comprese fra 0 e 63. L'intero concetto è meglio illustrato con una versione del programma originale del disegno 1:

```
10 V = 53248: POKE V+21,4 : POKE 2042,13
20 FOR N = 0 TO 62 : READ Q : POKE 832+N,Q : NEXT
25 POKE V+5, 100
30 FOR X = 0 TO 255
40 POKE V+4,X
50 NEXT
60 POKE V+16,4
```

```
70 FOR X = 0 TO 63
80 POKE V+4, X
90 NEXT
100 POKE V+16,0
110 GOTO 30
```

La riga 60 definisce il bit più significativo per il disegno 2. La riga 70 inizia 10 spostamento della locazione nella direzione standard X, spostando il disegno 2 del resto del percorso sullo schermo.
La riga 100 e importante in quanto "esclude" il MSB in modo che il disegno può iniziare di nuovo a spostarsi dal margine sinistro dello schermo.

ARITMETICA BINARIA

Va oltre gli scopi di questo manuale introduttivo entrare nei particolari del modo in cui il computer tratta i numeri. Forniremo comunque una buona base per la conoscenza del processo per consentire di realizzare sofisticati esempi di animazione. Prima di procedere occorre definire alcuni termini:

BIT - Si tratta della più piccola quantità di informazione che un computer può memorizzare.
Si pensi ad un bit come ad un interruttore che può essere "acceso" o "spento" (rispettivamente on e off). Quando un BIT e è "on" ha un valore di 1; quando un BIT è "off" ha un valore di 0.

Dopo il BIT il successivo livello e il BYTE.

BYTE - Il BYTE e definito come una serie di BIT. Dato che un BYTE e costituito da una serie di 8 BIT, e possibile avere in effetti un totale di 255 diverse combinazioni di BIT. In altre parole, e possibile avere tutti i BIT "off" in modo che il BYTE appaia come segue:

```
   128      64      32      16       8       4       2       1
+-------+-------+-------+-------+-------+-------+-------+-------+
|   0   |   0   |   0   |   0   |   0   |   0   |   0   |   0   |
+-------+-------+-------+-------+-------+-------+-------+-------+
```

Il suo valore sarà in questo caso 0. Tutti i BIT "on" appariranno come segue:

```
   128      64      32      16       8       4       2       1
+-------+-------+-------+-------+-------+-------+-------+-------+
|   1   |   1   |   1   |   1   |   1   |   1   |   1   |   1   |
+-------+-------+-------+-------+-------+-------+-------+-------+
```

il che equivale a dire 128 + 64 + 32 + 16 + 8 + 4 + 2 + 1 = 255.

La fase successiva è detta REGISTRO.

REGISTRO - È definito come un blocco di BYTE riuniti. Ma in questo caso ciascun REGISTRO è realmente lungo soltanto 1 BYTE. Una serie di REGISTRI costituisce una MAPPA DEI REGSTRI. Le mappe dei registri sono tabelle tipo quelle osservate nella creazione del disegno del pallone. Ciascun REGISTRO controlla una diversa funzione: a esempio quello che attiva il disegno è detto registro di abilitazione. Per rendere il disegno più lungo occorre espandere il registro X mentre Per renderlo più largo occorre espandere il registro Y. Tener presente che un REGISTRO è un BYTE che esegue un compito specifico.

Passiamo ora al resto dell'aritmetica binaria.

CONVERSIONE DA BINARIO A DECIMALE

```
+-----------------------------------------------------------------------+
|                             Valore Decimale                           |
+-------+-------+-------+-------+-------+-------+-------+-------+-------+
|  128  |   64  |   32  |   16  |   8   |   4   |   2   |   1   |       |
```

0	0	0	0	0	0	0	1	2^0
0	0	0	0	0	0	1	0	2^1
0	0	0	0	0	1	0	0	2^2
0	0	0	0	1	0	0	0	2^3
0	0	0	1	0	0	0	0	2^4
0	0	1	0	0	0	0	0	2^5
0	1	0	0	0	0	0	0	2^6
1	0	0	0	0	0	0	0	2^7

Usando la combinazione di tutti gli otto bit, è possibile ottenere qualsiasi valore decimale da 0 a 255. Si comincia ora a vedere perché quando si inseriscono (POKE) valori di carattere o di colore nelle locazioni di memoria i valori devono essere nel camp da 0 a 255? Ciascuna locazione di memoria può contenere un byte di informazione.

Qualsiasi combinazione possibile di otto 0 e 1 si trasforma in un valore decimale unico compreso tra 0 e 255. Se tutte le posizioni contengono un valore del byte è zero; "00000011" equivale a 3 e così via. Ciò costituisce la base per la creazione dei dati che rappresentano i disegni e per la loro manipolazione. A titolo di esempio, se questo raggruppamento di bit rappresentasse parte di un disegno (0 è uno spazio, 1 è un'area colorata)

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	1	1	1	1	1	1	1

128 + 64 + 32 + 16 + 8 + 4 + 2 + 1 = 255

Occorrerebbe inserire (POKE) 255 nell'appropriata locazione di memoria per rappresentare quella parte dell'oggetto.

SUGGERIMENTO:

Per risparmiare il fastidio di convertire i numeri binari in valori decimali - e occorrerà farlo spesso - basterà usare il seguente programma. È una buona idea immettere e salvare il programma per uso futuro.

```
5 REM BINARY TO DECIMAL CONVERTER
10 INPUT "ENTER 8-BIT BINARY NUMBER :";A$
12 IF LEN(A$) <> 8 THEN PRINT "8 BITS PLEASE..." : GOTO 10
15 TL = 0 : C = 0
20 FOR X = 8 TO 1 STEP -1 : C = C + 1
30 TL = TL + VAL(MID$(A$,C,1))*2^(X-1)
40 NEXT X
50 PRINT A$; " BINARY = "; TL ;" DECIMAL"
60 GOTO 10
```

Questo programma prende il numero binario che è stato immesso come stringa e ne osserva ciascun carattere da sinistra a destra (la funzione MID$). La variabile C indica su quale carattere lavorare man mano che il programma esegue le iterazioni.

La funzione VAL nella riga 30 dà il valore effettivo del carattere. Dato che si stanno trattando caratteri numerici, il valore è lo stesso del carattere. Per esempio, se il primo carattere di A$ è 1, il valore sarà a sua volta 1.

La parte finale della riga 30 moltiplica il valore del carattere corrente per l'appropriata potenza di 2. Dato che nell'esempio il primo valore è nella posizione 2|7, TL sarebbe la prima volta pari a 1 moltiplicato 128, ossia 128. Se il bit è 0 il valore per quella posizione sarebbe pure zero.

Questo processo viene ripetuto per tutti gli otto caratteri dato che TL tiene nota del valore decimale corrente del numero binario.

CAPITOLO 7

CREAZIONE DELLA MUSICA CON IL COMMODORE 64

- Struttura di un programma sonoro
- Melodie con il COMMODORE 64
- Definizione di suono
- Programmi dimostrativi

La maggior parte dei programmatori usa il suono del computer per due ragioni: fare della musica e generare effetti sonori. Prima di entrare nei dettagli della programmazione sonora, vediamo velocemente come è strutturato un programma sonoro... e includiamo anche un piccolo programma con il quale fare degli esperimenti.

STRUTTURA DI UN PROGRAMMA SONORO

Innanzitutto bisogna conoscere le 5 basi su cui generare il suono sul COMMODORE 64. Volume, controllo della forma d'onda, attack/decay, sustain/release (ADSR) e alta/bassa frequenza. La prima (attack/decay) è responsabile per la tonalità del suono con il quale è possibile distinguere i diversi strumenti musicali. Queste importanti caratteristiche si possono modificare anche da programma.

Per questo proposito il COMMODORE 64 è equipaggiato con un integrato sonoro per l'interfacciamento (SID). Il SID contiene due registri per ottenere i parametri per sintetizzare un suono specifico. Infatti, il COMMODORE 64 è capace di emettere tre suoni (voci) contemporaneamente. Vediamo la prima di queste voci.

L'indirizzo base del SID è 54272, abbreviamolo con un variabile, SI:

```
SI = 54272
```

La nota di una tonalità, o l'altezza, è data dalla sua frequenza che è determinata da un parametro registrato del 6 Questo parametro può avere un valore che va da quasi O a 65000. Nel capitolo precedente abbiamo visto, che numeri così grandi non possono essere immagazzinati su un singolo byte di memoria. Così dobbiamo separare il parametro di frequenza in due; ordine di byte basso, ordine di byte alto. Questa coppia di byte occupa i primi due registri del SID.

```
FL = SI         (frequenza, ordine di byte basso «low»).
FH = SI + 1     (frequenza, ordine di byte alto «high»).
```

Il volume può essere settato in 16 scale diverse in un raggio compreso tra O (spento) a 15 (volume massimo). I parametri corrispondenti sono memorizzati nel registro 24.

```
L = SI + 24     (volume)
```

Per selezionare una delle forme d'onda menzionate precedentemente «poke» uno di questi parametri: 17, 33, 65 o 129, dentro questo registro. Se si sceglie il 65 (impulso rettangolare) si deve definire un parametro addizionale per determinare il «duty cycle», che è la relazione tra «ON» e «OFF», della onda rettangolare. Entrambi i parametri sono memorizzati nei registri 2 e 3.

```
TL = SI + 2     (duty cycle, ordine di byte basso «low»)
TH = SI + 3     (duty cycle, ordine di byte alto «high»)
```

Vediamo insieme la più affascinante possibilità: la modulazione di una tonalità. Ci sono 4 diverse fasi che formano la struttura dell'impulso di un tono. Il primo è il tempo di attack. Se non si ha familiarità con questi concetti si può pensare per «attack» come la media di incremento con cui una nota raggiunge il suo volume massimo. La fase seguente è il «decay», che è la media con la quale la nota cade dal suo livello di volume più alto, al livello che noi chiamiamo «sustain» che è il terzo parametro che abbiamo specificato. Dopo la fase sustain che è il tempo nel quale si tiene premuto il tasto, la nota torna a 0, con una media di incremento che chiamiamo «release». Così abbiamo descritto i 4 parametri che caratterizzano la struttura di una nota. Ricapitolando i parametri sono ATTACK/DECAY/SUSTAIN/RELEASE or «ADSR». Ognuno di questi parametri può essere settato in 16 scale diverse. Ogni parametro ha bisogno di 4 bits per essere specificato. Questo significa che abbiamo bisogno di 2 registri diversi per ottenere tutti e 4 i parametri. Così definiamo:

```
A = SI + 5      (ATTACK/DECAY)
H = SI + 6      (SUSTAIN/RELEASE)
```

Ognuno di questi registri è diviso in due nibble (vedi paragrafo precedente per l'aritmetica binaria). Il parametro attack è determinato dai 4 bit più significativi del registro 5, mentre per determinare il decay sono usati i 4 bit meno significativi, dello stesso registro.

Il livello di sustain è tenuto nei 4 bit più significativi del registro 6, mentre il release è memorizzato nei meno significativi del medesimo.

Piccoli valori nella parte di attack del registro 5, causa un suono molto duro; mentre un valore grande (fino a 15) produce un suono debole.

Esaminiamo tutti i parametri fin qui discussi, usando un piccolo programma dimostrativo.

PROGRAMMA DIMOSTRATIVO

Prima di tutto bisogna decidere quale voce vogliamo usare. Per ognuna di queste dobbiamo abilitare i 4 parametri prima menzionati (volume, forma d'onda ecc.). Si possono usare fino a tre voci simultaneamente; nel nostro caso useremo soltanto la voce numero 1.

```
10 SI=54272: FL=SI:         1. Definizione dei registri
   FH=SI+1: W=SI+4:
   A=SI+5: H=SI+6:
   L=SI+24
20 POKE L,15                2. Volume massimo
30 POKE A,16+9              3. Attack
40 POKE H,4*16+4            4. Sustain - Release
50 POKE FH,29:POKE FL,69    5. Frequenza High e Low, per la definizione della nota
a (la).
                            Vedere l'appendice P per la definizione delle diverse
note
60 POKE W,17                6. Generatore di forma d'onda in posizione ON
70 FORT=1TO500:NEXT         7. Iterazione che determina la durata della nota
80 POKE W,0:POKE A,0:       8. Azzeramento della forma d'onda
   POKE H,0
```

Dopo aver battuto «run» si può udire la nota generata da questo programma.

MELODIE CON IL COMMODORE 64

Non c'è bisogno di essere un musicista per comporre delle melodie con il COMMODORE 64. Il prossimo programma è un esempio di come si può fare. Useremo sempre una voce delle tre a disposizione.

Cancellare il programma precedente con NEW e memorizzare il seguente:

```
   10 REM TONALITÀ                    Nome del programma
   20 SI=54272:FL=SI:FH=SI+1          Definizione dei registri
      E=Sl+4:A=SI+5:H=SI+6:
      L=SI+24
   30 POKE L,15                       Volume massimo
   40 POKE A,9                        Attack
   50 READ X:READ Y                   Frequenza del Low-bit e High-bit in relazione ai
dati delle linee 130 e 140
   60 IFY=-1 THEN POKE W,0:END        Se il programma trova -1; termina
   70 POKE FH,X:POKE FL,Y             Poke nei registri di frequenza Low-bit High-bit
   80 POKE W,17                       Generatore dl forma d'onda in posizione ON
   90 FORT= 1TO100:NEXT               Iterazione di ritardo
   100 POKE W,0                       Generatore in posizione OFF
   110 FORT= 1T050:NEXT               Iterazione di ritardo per RELEASE
   120 GOTO40                         TONO seguente
   130 DATA17,103,19,137,21,237,      Questi numeri determinano le note delle
   23,59,26,20,29,69,32,219,34,201    scale DO-DURO
   140 DATA1,-1                       Questi dati, che non hanno significato come
frequenza,
                                      segnalano alla linea 60 che il programma è finito
```

Se vogliamo generare un suono come quello del cembalo, dobbiamo cambiare la linea 80 nel modo seguente:

80 POKE W,33

Questo poke seleziona un dente di sega come forma d'onda, che contiene molte armoniche per generare un suono tagliente; ma la scelta della forma d'onda è soltanto un modo per cambiare un suono. Con un parametro speciale nei registri ADSR possiamo cambiare il cembalo in un banjo. Questo si ottiene con il seguente comando POKE nella linea40:

40 POKE A,3

Possiamo anche imitare il suono di diversi strumenti musicali come un vero sintetizzatore. Come fare? Meglio dire in che modo cambiare il contenuto dei registri; cosa che vedremo adesso.

IMPORTANTE PER LA DEFINIZIONE DEL SUONO

1. Volume - la scelta del volume è valida per tutti e tre i generatori di suono. Il corrispondente registro si troverà all'indirizzo 54296. Il volume massimo si otterrà con una poke pari a 15

POKE L,15 oppure POKE 54296,15

Per annullare il generatore di suono basta una poke pari a 0 in questo registro.

POKE L,0 oppure POKE 54296,0

Di solito si abilita il volume all'inizio di un programma musicale; ma attraverso l'alterazione programmata di questi parametri, si possono creare effetti interessanti.

2. Forma d'onda - come abbiamo visto nell'esempio precedente, la forma d'onda influenza il carattere sonoro di una tonalità. Per ognuna di queste voci si può scegliere una forma d'onda diversa. Questa può essere triangolare, a dente di sega, rettangolare e rumore bianco. La tavola in basso contiene i diversi indirizzi dei

registri e i loro contenuti. Se si vuole scegliere per la prima voce, la forma d'onda triangolare, eseguire i seguenti comandi:

POKE W,17 oppure POKE 54276,17

Il primo numero (indirizzo) indica il registro, mentre il secondo numero (contenuti dell'indirizzo del registro) controlla la forma d'onda.

DEFINIZIONE DELLA FORMA D'ONDA

Registri				Contenuti			
Voce	1	2	3	Rumore	Rettangolare	Dente di Sega	Triangolare
	4	11	18	129	65	33	17

Questa è la tavola usata nella linea 30 del programma «scala musicale». Con POKE SI+4,17 si è scelta la forma d'onda triangolare, che si è trasformata in dente di sega cambiando il 17 con il 33. Vediamo adesso come può cambiare la struttura che determina la media di incremento del volume, in una tonalità. Da ricordare che questa nota musicale si ottiene soltanto avendo predisposto sia il volume che la forma d'onda.

3. ADSR - il valore per ATTACK e DECAY, come per la forma d'onda, può essere scelto separatamente per ogni voce ed è rappresentato da un numero. ATTACK è la media di incremento con cui la nota raggiunge il suo volume massimo. DECAY determina il valore del tempo con cui la nota cade dal suo volume massimo al livello di SUSTAIN programmato. Se si seleziona 0 come livello di SUSTAIN, il parametro DECAY determina il tempo del livello di volume; cosicché è identico alla durata della nota. Il valore degli indirizzi per le differenti voci e la predisposizione di AD può essere vista nella seguente tavola, I valori per ATTACK e DECAY sono semplicemente addizionati e pokati nei registri prestabiliti.

ATTACK-DECAY

Registri				Contenuti	
Voce	1	2	3	Attack	Decay
	5	12	19	15*16 (Debole) … 0*16 (Duro)	15 Debole … 0 Duro

Se si seleziona soltanto un tempo di ATTACK con una poke 54277,64, il tempo di DECAY si setta automaticamente a 0, e viceversa. Con poke 54277,66 si setta il valore medio di ATTACK (64=4*16) e DECAY ad un valore piccolo (2); quindi 66 è il risultato della somma 64+2. Il miglior modo per riconoscere un parametro così composito è di scrivere POKE A,4*16+2, invece di POKE 54277,66 (l'indirizzo del registro A deve essere specificato prima). Abbiamo quindi raggiunto il punto dove possiamo provare le cose fin qui descritte con un piccolo programma. Come al solito battiamo NEW, premiamo return, e inseriamo il seguente programma:

```
10 REM PROGRAMMA SPERIMENTALE
20SI=54272: FL=SI: FH=SI+1: TL=SI+2:
TH= SI+3: W=SI+4: A=SI+5: H=SI+6:
L= SI+24
30 PRINT"PREMI UN TASTO"                Commento video
40 GETZ$:IFZ$=""THEN40                  Chiave premuta?
50 POKE L,15                            Volume
60 POKE A,1*16+5                        Attack-Decay
70 POKE H,0*16+0                        Sustain-Release
80 POKE TH,8: POKE TL,0                 Duty cycle
90 POKE FH,14: POKE FL,162              Frequenza
100 POKE W,17                           Generatore, forma d'onda ON
110 FORT= 1TO200:NEXT                   Iterazione
```

```
120 POKE W,0                            Generatore OFF
130 GOTO40                              Premi un tasto
```

Abbiamo usato la voce 1 per creare il tono con un corto ATTACK e un corto DECAY, dopo aver raggiunto il massimo volume (linea 60). L'effetto sonoro ricavato è un suono metallico. Per creare un suono diverso è sufficiente cambiare questa riga. Per far questo premere il tasto [RUN/STOP] e listare il programma. Dopo averlo visualizzato sullo schermo cambiare la linea 60 nel seguente modo:

60 POKE A,11*16+14

Premendo return il computer memorizza la nuova riga nel programma in memoria.

Il tono, che si otterrà con questa predisposizione, è come quelle di un oboe o di un flauto. Facciamo alcune prove e cambiamenti alla forma d'onda e all'ADRS, per renderci conto di come le diverse predisposizioni di questi parametri influenzano il carattere di un tono.

Usando il parametro SUSTAIN si può determinare il volume di un tono, dopo la fase di ATTACK/DECAY. Di solito la durata di un tono è determinata usando il loop FOR … NEXT. Come nel registro precedente, SUSTAIN e RELEASE sono determinati dallo stesso registro; quindi non si deve fare altro che addizionare il valore di entrambi i parametri e poke la somma di questi, nel registro corrispondente. Vedi tabella in basso.

SUSTAIN-RELEASE

Registri				Contenuti	
Voce	1	2	3	Sustain	Release
	6	13	20	15*16 (Alto) … 0*16 (Basso)	15 Lento … 0 Veloce

Cambiando il valore degli 0 nelle righe di programma precedente, tra i valori 0 - 15, si può udire la differenza.

4. Selezione di voci e note - come visto precedentemente si devono specificare i 2 valori, per determinare la frequenza o la nota di un singolo tono. Chiameremo questi valori, ordine alto di byte e ordine basso di byte, della frequenza. L'assegnazione del nome della nota per il valore della frequenza è riportata nella tabella dell'appendice P.

Siccome le voci sono relegate a indirizzi diversi, con il COMMODORE 64 si possono programmare le 3 voci indipendentemente; in questo modo si possono creare canzoni a tre voci o corde.

INDIRIZZI DEL GENERATORE DI NOTA E POKE PER HIGH BYTE - LOW BYTE PER NOTE IN 5. OTTAVA

Registri				Contenuti per le note in 5. ottava												
Voce	1	2	3	C	C#	D	D#	E	F	F#	G	G#	A#	H#	H	C
HIGH	1	8	15	35	37	39	41	44	46	49	52	55	58	62	66	70
LOW	0	7	14	3	24	77	163	29	188	132	117	148	226	98	24	6

Per generare una C (DO) con la VOCE i POKE le seguenti istruzioni:

POKE 54273,35: POKE 54272,3 oppure POKE SI+1,35: POKE SI,3

Ugualmente la VOCE 2 ai ottiene con:

POKE 54280,35: POKE 54279,3 oppure POKE SI+8,35: POKE SI+7,3

ESECUZIONE DI UN MOTIVO SUL COMMODORE 64

Il seguente programma può essere usato per comporre o suonare un motivo (usando VOCE 1). Ci sono due importanti lezioni in questo programma: innanzitutto come abbreviare tutti i lunghi numeri di controllo nella prima

riga del programma dopodiché è possibile usare la lettera W per «forma d'onda» invece del numero 54276.

La seconda lezione riguarda il modo in cui si usano i dati. Questo programma è scritto in modo da consentire di immettere tre numeri per ciascuna nota: il VALORE DELLA NOTA IN HIGH FREQUENCY, il VALORE DELLA NOTA IN LOW FREQUENCY e la DURATA DELLA NOTA CHE VERRA' SUONATA.

Per questo motivo si userà una durata di 125 per una croma, 150 per una semiminima, 375 per una semiminima puntata, 500 per una minima e 1000 per una semibreve. Questi valori possono essere aumentati o diminuiti per adeguarli ad un particolare tempo o al proprio gusto musicale.

Per vedere come viene immesso un motivo, osservare la riga 100. Sono stati immessi 34 e 75 come regolazioni di HIGH e LOW FREQUENCY per suonare un «Do» (dalla scala campione indicata in precedenza) e quindi il numero 250 per la semiminima. Così la prima nota del nostro motivo è una nota Do semiminima. Anche la seconda nota è una semiminima, ma stavolta è un Mi... e così via fino alla fine del motivo. E' possibile immettere qualsiasi altro motivo in questo modo, aggiungendo tante righe di istruzioni DATA quante ne occorrono. E' possibile continuare i numeri delle note e di durata da una riga alla successiva ma ciascuna riga deve cominciare con la parola DATA. DATA-1,-1,-1 deve essere l'ultima riga nel programma. Questa riga «termina» il motivo.

Battere la parola NEW per cancellare il precedente programma e battere il programma seguente, quindi battere RUN per ascoltare il motivo.

MICHAEL ROW THE BOAT ASHORE-1 MISURA

```
5 V=54296:W=54276:A=54277:H=54273:LF=54272:S=54278:PH=54275:PL=54274
10 POKEV,15:POKEW,65:POKEA,190:POKEH,15:POKEL,15
20 READH
30 READL
40 READD
50 IFH=-1THENEND
60 POKEHF,H:POKELF,L
70 FORX=D-50TOD-20:POKES,136:NEXT
80 FORT=1TOD:NEXT:POKEHF,0:POKELF,0:POKEW,0
90 GOTO10
100 DATA34,75,250,43,52,250,51,97,375,43,52,125,51,97
105 DATA250,57,172,250
110 DATA51,97,500,0,0,125,43,52,250,51,97,250,57,172
115 DATA1000,51,97,500
120 DATA-1,-1,-1
```

CREAZIONE DI EFFETTI SONORI

A differenza della musica, gli effetti sonori sono spesso legati ad una specifica «azione» di programmazione ad esempio l'esplosione creata da un combattente spaziale quando penetra attraverso una barriera nel gioco «Guerre spaziali» … o la cicalina in un programma gestionale che dice all'utente che sta per cancellare per errore il suo disco.

Sono disponibili numerose opzioni se si vogliono creare diversi effetti sonori. Ecco 10 idee di programmazione che potrebbero aiutare ad iniziare la sperimentazione con gli effetti sonori:

1. Cambio del volume mentre viene eseguita la nota, ad esempio per creare un effetto d'eco.
2. Alternanza rapida tra due note per creare un effetto «tremolo».
3. Forma d'onda: provare diverse regolazioni per ciascuna voce.
4. Attack/decay: per alterare la velocità di salita e di caduta di un suono.
5. Sustain/release: per prolungare o smorzare il volume di un effetto sonoro oppure per combinare una serie di suoni. Occorre provare diverse regolazioni.
6. Effetti a più voci… cioè l'esecuzione di più di una voce contemporaneamente, con ciascuna voce controllata indipendentemente o con una voce che suona più o meno a lungo di un'altra o che serve da «eco» o risposta ad una prima nota.
7. Cambio delle note sulla scala o cambio dell'ottava usando i valori nella tabella VALORI DELLE NOTE MUSICALI.
8. Uso della forma d'onda quadra e di diverse regolazioni di impulsi per creare effetti diversi.
9. Uso della forma d'onda di rumore per generare «rumore bianco» per accentuare gli effetti sonori tonali o

per creare esplosioni, colpi di cannone o rumore di passi. Le stesse note musicali che creano la musica possono anche essere usate con la forma d'onda di rumore per creare diversi tipi di rumore bianco.

10. Combinazione di parecchie frequenze HIGH/LOW in rapida successione su diverse ottave.
11. Filtrazione: provare la regolazione POKE extra nell'Appendice M.

ESEMPI DI EFFETTI SONORI DA PROVARE

I seguenti programmi possono essere aggiunti a pressoché qualsiasi programma BASIC e forniscono suggerimenti di programmazione intesi a dimostrare le possibilità nel campo degli effetti sonori di COMMODORE 64.

```
10 REM
20 Sl=54272:FL=SI:FH=SI+1:TL=SI+2:TH=SI+3:W=SI+4:A=SI+5:H=SI+6:L=SI+24
30 POKEL15:POKETH,15:POKETL,15:POKEA,0*16+0:POKEH,15*16
40 POKEW65
50 FOR X=250TO0 STEP-2:POKEFH,40:POKEFL,X:NEXT
60 FOR X=150TO0STEP-4POKEFH,40:POKEFL,X:NEXT
70 POKEW,0
```

```
10 REM SPARO Dl UNA PISTOLA
20 SI=54272:FL=SI:FH=SI+1:TL=SI+2:TH=SI+3:W=SI+4:A=SI+5:H=SI+6:L=SI+24
30 FORX=15TO0STEP-1
40 POKEL,X:POKEA,15:POKEH,0:POKEFH,40:POKEFL,200:POKEW,129
50 NEXT
60 POKEW,0:POKEA,0
```

```
10 REM MOTORE
20 SI=54272
30 FORK=0TO24:READX: POKESI+K,X:NEXT
40 DATA 9,2,0,3,0,0,240
50 DATA 12
60 DATA 16,2,0,6,0,0,64
70 DATA 0,30,243,31:REM FILTER
80 POKESI+4,65:POKESI+11,65:POKESI+18,65
```

CAPITOLO 8

MANIPOLAZIONE AVANZATA DEI DATI

- READ e DATA
- Medie
- Variabili con indice
  Matrici unidimensionali
  Un ripasso delle medie
- DIMENSION
- Lancio di dadi simulato con le matrici
- Matrici bidimensionali

READ e DATA

Si è visto come assegnare i valori alle variabili direttamente nell'ambito del programma (A=2) e come assegnare diversi valori mentre il programma è in esecuzione - attraverso l'istruzione INPUT.

Ci sono comunque dei casi in cui nessuno di questi modi è adatto allo scopo, specialmente se sono coinvolte numerose informazioni.

Provare questo breve programma:

```
10 READ X
20 PRINT "X ORA VALE: ";X
30 GOTO 10
40 DATA 1,34,10.5,16,234.56

RUN

X ORA VALE: 1
X ORA VALE: 34
X ORA VALE: 10.5
X ORA VALE: 16
X ORA VALE: 234.56

?OUT OF DATA ERROR IN 10
READY
```

Nella riga 10 il computer legge (READ) un valore dall'istruzione DATA ed assegna quel valore a X. Ad ogni iterazione viene letto il successivo valore nell'istruzione DATA e quel valore viene assegnato a X e stampato (PRINT). Un puntatore nel computer tiene nota di quale valore deve essere usato successivamente:

```
          +--------PUNTATORE
          |
40 DATA 1,34,10.5,16,234.56
```

Quando tutti i valori sono stati usati e il computer esegue di nuovo l'iterazione cercando un altro valore, viene visualizzato l'errore OUT OF DATA (mancanza di dati) in quanto non ci sono altri valori da leggere (READ).

È importante seguire con precisione il formato dell'istruzione DATA:

```
40 DATA 1,34,10.5,16,234.56 <--- Nessuna virgola
         |
         +--- La virgola separa ogni elemento
```

Le istruzioni DATA possono contenere numeri interi, numeri reali (234.56), o numeri espressi in notazione scientifica. Ma non è possibile leggere (READ) altre variabili o avere operazioni aritmetiche nelle righe DATA. Ciò sarebbe scorretto:

```
40 DATA A,23/56,2*5
```

È possibile comunque usare una variabile stringa in un'istruzione READ e quindi inserire l'informazione stringa nella riga DATA. Quanto segue è accettabile:

```
NEW

10 FOR X = 1 TO 3
15 READ A$
20 PRINT "A$ ORA VALE: ";A$
30 NEXT
40 DATA THIS, IS, FUN
```

```
RUN

A$ IS NOW: THIS
A$ IS NOW: IS
A$ IS NOW: FUN
READY
```

Notare che stavolta l'istruzione READ è stata posta all'interno di una iterazione FOR...NEXT. Questa iterazione è stata quindi eseguita fino ad uguagliare il numero dei valori delle istruzioni DATA.

In molti casi cambierà il numero di valori nell'istruzione DATA ogni volta che il programma viene eseguito. Un modo per evitare di contare il numero dei valori ed evitare un errore OUT OF DATA consiste nell'inserire un «flag» come ultimo valore nella riga DATA, ossia un valore che i dati non eguaglierebbero mai, ad esempio un numero non negativo o un numero molto grande o molto piccolo. Quando il valore viene letto (READ) il programma passa alla parte successiva.

C'è un modo per riutilizzare gli stessi DATA successivamente nel programma ripristinando (RESTORE) il puntatore di dati all'inizio della lista relativa.

Si aggiunga la riga 50 al precedente programma:

```
50 GOTO 10
```

Si ottiene sempre l'errore OUT OF DATA in quanto come il programma salta all'indietro per leggere i dati, il puntatore indica che tutti i dati sono stati usati. Si aggiunga ora:

```
45 RESTORE
```

e si esegua (RUN) il programma di nuovo. Il puntatore è stato ripristinato (RESTORE) ed i dati possono essere letti (READ) in continuazione.

MEDIE

Il seguente programma illustra l'uso pratico di READ e DATA per leggere una serie di numeri e calcolarne la media.

```
NEW

5 T = 0 : CT = 0
10 READ X
20 IF X = -1 THEN 50 : REM CHECK FOR FLAG
25 CT = CT + 1
30 T= T + X : REM UPDATE TOTAL
40 GOTO 10
50 PRINT "THERE WERE"; CT ;"VALUES READ"
60 PRINT "TOTAL ="; T
70 PRINT "AVERAGE ="; T/CT
80 DATA 75, 80, 62, 91, 87, 93, 78, -1

RUN
THERE WERE 7 VALUES READ
TOTAL = 566
AVERAGE = 80.8571429
```

La riga 5 imposta CT, il ConTatore e T il Totale uguale a zero. La riga 10 legge (READ) un valore e assegna quel valore a X. La riga 20 controlla se il valore corrisponde al flag (in questo caso -1). Se il valore letto (READ) è una parte dei DATA validi, CT è incrementato di 1 e X viene sommato al totale.

Quando il flag viene letto (READ), il programma salta alla linea 50 che stampa (PRINT) il numero dei valori letti.

La riga 60 stampa (PRINT) il totale e la riga 70 divide il totale per il numero dei valori per ottenere la media.

Usando un flag al termine dei DATA è possibile inserire qualsiasi numero di valori nelle istruzioni DATA - che

possono estendersi su parecchie righe - senza preoccuparsi di contare il numero dei valori immessi.

Un'altra variazione dell'istruzione READ comporta l'assegnazione di informazioni della stessa riga DATA a diverse variabili. Queste informazioni possono essere addirittura una miscela di dati stringa e di valori numerici. È possibile fare tutto ciò nel programma seguente che leggerà un nome, alcuni punteggi - ad esempio di bowling - e stamperà i nomi, i

punteggi ed il punteggio medio:

```
NEW

10 READ N$, A, B, C
20 PRINT N$;"'S SCORES WERE :"; A ;" "; B ;" "; C
30 PRINT "AND THE AVERAGE IS :";(A+B+C)/3
40 PRINT : GOTO 10
50 DATA MIKE, 190, 185, 165, DICK, 225, 245, 190
60 DATA JOHN, 155, 185, 205, PAUL, 160, 179, 187

RUN
MIKE'S SCORES WERE : 190   185   165
AND THE AVERAGE IS : 180

DICK'S SCORES WERE : 225   245   190
AND THE AVERAGE IS : 220
```

Nell'esecuzione del programma le istruzioni DATA sono state nello stesso ordine in cui l'istruzione READ aspettava le informazioni: un nome (una stringa) quindi tre valori. In altre parole N$ la prima volta ottiene i DATA «MIKE», A in READ corrisponde a 190 nell'istruzione di dati, «B» a 185 e «C» a 165. Il processo viene quindi ripetuto nello stesso ordine per il resto delle informazioni (Mike e i suoi punteggi, Dick e i suoi punteggi, etc.).

VARIABILI CON INDICE

In passato, per rappresentare valori, sono state usate soltanto variabili BASIC semplici, ad esempio A, A$ e NU.

Queste erano costituite da una singola lettera seguita da una singola lettera o cifra. In uno qualsiasi dei programmi che si potranno scrivere non si avrà mai bisogno di nomi variabili in numero tali da superare tutte le combinazioni possibili di lettere e numeri. Si è invece limitati al modo in cui le variabili sono usate con i programmi.

Verrà ora introdotto il concetto di variabili con indice.

```
A(1)
| |
| +----- INDICE
|
+------- VARIABILE
```

L'espressione si legge: A indice 1. Una variabile con indice si compone di una lettera seguita da un indice racchiuso tra parentesi. Notare la differenza tra A, A1 e A(1). Ciascuna espressione è diversa dalle altre ma soltanto A(1) è una variabile con indice.

Le variabili con indice, come le variabili semplici, individuano una locazione di memoria nell'ambito del computer. Si può pensare alle variabili con indice come a scatole per memorizzare le informazioni, esattamente come per le variabili semplici:

```
        +-------------------------+
A(0)    |                         |
        +-------------------------+
A(1)    |                         |
        +-------------------------+
A(2)    |                         |
        +-------------------------+
```

```
  A(3)   |                         |
         +-------------------------+
  A(4)   |                         |
         +-------------------------+
```

Se si scrivesse:

10 A(0) = 25: A(3) = 55: A(4) = 45.3

la memoria apparirebbe come segue:

```
         +-------------------------+
  A(0)   | 25                      |
         +-------------------------+
  A(1)   |                         |
         +-------------------------+
  A(2)   |                         |
         +-------------------------+
  A(3)   | 55                      |
         +-------------------------+
  A(4)   | -45.3                   |
         +-------------------------+
```

Questo gruppo di variabili con indice è anche detto matrice. In questo caso si tratta di una matrice unidimensionale. Successivamente verranno presentate le matrici multidimensionali.
Gli indici possono anche essere più complessi per includere variabili o calcoli. Quelle che seguono sono variabili con indice:

```
A(X)      A(X+1)      A(2+1)      A(1*3)
```

Le espressioni fra parentesi sono valutate secondo le stesse regole delle operazioni aritmetiche indicate nel capitolo 2.
Ora che sono state chiarite le regole di base, come è possibile mettere all'opera le variabili con indice? Un modo consiste nel memorizzare una lista di numeri immessi con le istruzioni INPUT o READ.
Si useranno le variabili con indice per eseguire le medie in un modo diverso.

```
5 PRINT CHR$(147)
10 INPUT "HOW MANY NUMBERS :"; X
20 FOR A = 1 TO X
30 PRINT "ENTER VALUE #"; A ;: INPUT B(A)
40 NEXT
50 SU = 0
60 FOR A = 1 TO X
70 SU = SU + B(A)
80 NEXT
90 PRINT : PRINT "AVERAGE ="; SU/X

RUN
HOW MANY NUMBERS :? 5
ENTER VALUE # 1 ? 125
ENTER VALUE # 2 ? 167
ENTER VALUE # 3 ? 189
ENTER VALUE # 4 ? 167
ENTER VALUE # 5 ? 158

AVERAGE = 161.2
```

Esiste certamente un modo più facile per eseguire ciò che è stato fatto in questo programma ma esso illustra come funzionano le variabili con indice. La riga 10 chiede quanti numeri verranno immessi. Questa variabile, X, agisce

come contatore per l'iterazione nell'ambito della quale i valori vengono immessi ed assegnati alla variabile con indice, B.

Ogniqualvolta viene eseguita l'interazione INPUT, A viene aumentato di 1 e così il successivo valore immesso è assegnato al successivo elemento nella matrice A. Per esempio, dopo la prima iterazione A = 1, il primo valore immesso è assegnato a B(1). La successiva volta, A = 2, il successivo valore viene assegnato a B(2) così fino a che tutti i valori non sono stati immessi.

Ma ora entra in gioco una grande differenza. Una volta che tutti i valori sono stati immessi, questi vengono memorizzati nella matrice, pronti per essere utilizzati in numerosi modi. Prima veniva conservato un totale ogniqualvolta veniva eseguita l'iterazione INPUT o READ, ma non si poteva risalire ai singoli elementi di dati senza rileggere le informazioni.

Nelle righe da 50 a 80, è stata disegnata un'altra iterazione per sommare i vari elementi della matrice e quindi visualizzarne la media. Questa parte separata del programma mostra che tutti i valori sono memorizzati e che ad essi è possibile accedere come necessario.

Per dimostrare che tutti i singoli valori sono effettivamente memorizzati separatamente in una matrice, battere quanto segue immediatamente dopo aver eseguito il precedente programma:

```
FOR A = 1 TO 5: ?B(A),: NEXT

125      167      189      167
158
```

Lo schermo visualizzerà i valori effettivi quando i contenuti della matrice verranno stampati (PRINT).

DIMENSION

Se si tentasse di immettere pi- di 10 numeri nel precedente esempio, si otterrebbe un errore DIMENSION. Possono essere usate matrici con un massimo di 11 elementi (indici da 0 a 10 per una matrice unidimensionale) dove necessario esattamente come le variabili semplici possono essere usate ovunque all'interno di un programma. Le matrici di più di undici elementi devono essere «dichiarate» in un'istruzione di dimensione.

Aggiungere questa riga al programma:

```
5 DIM B(100)
```

Ciò fa sapere al computer che nella matrice si avrà un massimo di 101 elementi.

L'istruzione di dimensione può essere anche usata con una variabile cosicché, la riga seguente potrebbe sostituire la riga 5 (non dimenticarsi di eliminare la riga 5):

```
15 DIM B(X)
```

Ciò dimensionerebbe la matrice con il numero esatto di valori immessi.

Bisogna comunque fare attenzione. Una volta dimensionata, una matrice non può essere ridimensionata in un'altra parte del programma. È possibile per contro avere matrici multiple nell'ambito del programma e dimensionarle nella stessa riga come nell'esempio che segue:

```
10 DIM C(20), D(50), E(40)
```

LANCIO DI DADI SIMULATO CON LE MATRICI

Man mano che i programmi si fanno più complessi, l'uso delle variabili con indice riduce il numero delle istruzioni necessarie e rende più facile la scrittura del programma.

Può essere usata un'unica variabile con indice, ad esempio, per tener nota del numero di volte che compare una particolare faccia del dado:

```
10 INPUT "HOW MANY ROLLS: "; X
20 FOR L = 1 TO X
30 R = INT(6*RND(1))+1
40 F(R) = F(R) + 1
50 NEXT L
60 PRINT "FACE", "NUMBER OF TIMES"
70 FOR C=1 TO 6 : PRINT C, F(C) : NEXT
```

La matrice F, per Faccia, verrà usata per tener nota di quante volte compare una particolare faccia del dado. Per esempio, ogni volta che esce un 2, F(2) viene aumentato di uno. Usando lo stesso elemento della matrice per contenere il numero effettivo sulla faccia uscita si è eliminata la necessità di altre 5 variabili (una per ciascuna faccia) e numerose istruzioni per controllare quale è il numero uscito.

La riga 10 chiede quanti lanci si vogliono simulare.

La riga 20 stabilisce l'iterazione per eseguire il lancio casuale ed incrementare l'appropriato elemento della matrice di uno per ogni lancio.

Al termine di tutti i lanci, la riga 60 stampa (PRINT) il titolo e la riga 70 stampa (PRINT) il numero di volte che è comparsa ciascuna faccia.

Un semplice lancio potrebbe comparire come segue:

```
HOW MANY ROLLS: ? 1000
FACE       NUMBER OF TIMES
1          148
2          176
3          178
4          166
5          163
6          169
```

Unicamente a titolo di confronto, quello che segue è un modo per riscrivere lo stesso programma senza utilizzare le variabili con indice. Non preoccuparsi di batterlo ma notare invece le ulteriori istruzioni necessarie.

```
10 INPUT "HOW MANY ROLLS: "; X
20 FOR L = 1 TO X
30 R = INT(6*RND(1))+1
40 IF R = 1 THEN F1 = F1 + 1 : NEXT
41 IF R = 2 THEN F2 = F2 + 1 : NEXT
42 IF R = 3 THEN F3 = F3 + 1 : NEXT
43 IF R = 4 THEN F4 = F4 + 1 : NEXT
44 IF R = 5 THEN F5 = F5 + 1 : NEXT
45 IF R = 6 THEN F6 = F6 + 1 : NEXT
60 PRINT "FACE", "NUMBER OF TIMES"
70 PRINT 1, F1
71 PRINT 2, F2
72 PRINT 3, F3
73 PRINT 4, F4
74 PRINT 5, F5
75 PRINT 6, F6
```

Il programma è raddoppiato di dimensioni passando da 8 a 16 righe. Nei programmi più grandi i risparmi di spazio conseguiti usando le variabili con indice sono ancora maggiori.

MATRICI BIDIMENSIONALI

All'inizio di questo capitolo si sono conosciute le matrici unidimensionali, visualizzate come gruppi di scatole consecutive nell'ambito della memoria, ciascuna delle quali conteneva un elemento della matrice. Come dovrebbe apparire una matrice bidimensionale?

Innanzitutto una matrice bidimensionale verrebbe scritta nel modo seguente:

```
A(4,6)
| | |
| +-+--- Indici
|
+------- Nome Matrice
```

e potrebbe essere rappresentata come una griglia bidimensionale nell'ambito della memoria:

```
         0       1       2       3       4       5       6
     +-------+-------+-------+-------+-------+-------+-------+
0    |       |       |       |       |       |       |       |
     +-------+-------+-------+-------+-------+-------+-------+
1    |       |       |       |       |       |       |       |
     +-------+-------+-------+-------+-------+-------+-------+
2    |       |       |       |       |       |       |       |
     +-------+-------+-------+-------+-------+-------+-------+
3    |       |       |       |       |       |       |       |
     +-------+-------+-------+-------+-------+-------+-------+
4    |       |       |       |       |       |       |       |
     +-------+-------+-------+-------+-------+-------+-------+
```

Gli indici potrebbero indicare la fila e la colonna nell'ambito della tabella dove è caricato quel particolare elemento della matrice.

```
A(3,4) = 255
  | |
  | +--- Colonna
  |
  +----- Fila
```

```
         0       1       2       3       4       5       6
     +-------+-------+-------+-------+-------+-------+-------+
0    |       |       |       |       |       |       |       |
     +-------+-------+-------+-------+-------+-------+-------+
1    |       |       |       |       |       |       |       |
     +-------+-------+-------+-------+-------+-------+-------+
2    |       |       |       |       |       |       |       |
     +-------+-------+-------+-------+-------+-------+-------+
3    |       |       |       |       |  255  |       |       |
     +-------+-------+-------+-------+-------+-------+-------+
4    |       |       |       |       |       |       |       |
     +-------+-------+-------+-------+-------+-------+-------+
```

Se si assegnasse il valore 255 a A(3,4), 255 potrebbe essere visto come disposto nella quarta colonna della terza fila nell'ambito della tabella.

Le matrici bidimensionali si comportano secondo le stesse regole stabilite per quelle unidimensionali:

Devono essere dimensionate:	DIM A(20,20)
Assegnazione di dati:	A(1,1) = 255
Assegnazione di valori e altre variabili:	AB = A(1,1)
Valori PRINT:	PRINT A(1,1)

Se le matrici bidimensionali funzionano come le loro controparti più piccole, quali ulteriori capacità potranno affrontare le matrici ampliate?

Ad esempio, è possibile pensare ad un modo di usare una matrice bidimensionale per tabulare i risultati di un questionario per il club che comporta quattro domande ed elenca tre risposte possibili per ciascuna domanda? Il problema potrebbe essere rappresentato come segue:

```
QUESTIONARIO DEL CLUB

D1: SIETE A FAVORE DELLA RISOLUZIONE N.1?

[] 1-SI'        [] 2-NO        [] 3-INDECISO    ...e così via
```

La tabella delle matrici per questo problema potrebbe essere rappresentata come segue:

```
                                       RISPOSTE

                         SI                NO              INDECISO

                +----------------+----------------+----------------+
DOMANDA 1       |                |                |                |
                +----------------+----------------+----------------+
DOMANDA 2       |                |                |                |
                +----------------+----------------+----------------+
DOMANDA 3       |                |                |                |
                +----------------+----------------+----------------+
DOMANDA 4       |                |                |                |
                +----------------+----------------+----------------+
```

Il programma per eseguire la tabulazione effettiva del questionario dovrebbe apparire come quello mostrato a pagina 104.

Questo programma fa uso di molte delle tecniche della programmazione finora presentate. Anche se per il momento non c'è alcuna necessità di un programma del genere, vale la pena di seguirlo per vedere come funziona.

Il cuore del programma è una matrice bidimensionale 4 x 3, A(4,3). I risultati totali per ciascuna possibile risposta a ciascuna domanda sono contenuti nell'appropriato elemento della matrice. Per semplicità, non si useranno le prime file e colonne (da A(0,0) a A(0,4)). Ricordarsi comunque che quegli elementi sono sempre presenti nella matrice che viene disegnata.

In pratica se alla domanda uno risponde SI, A(1,1) verrebbe incrementato di uno - fila uno per la domanda uno e colonna uno per una risposta SI. Il resto delle domande e delle risposte seguono lo stesso profilo. Una risposta NO per la domanda tre aggiungerebbe uno all'elemento A(3,2) e così via.

```
 20 PRINT "{CLR/HOME}"
 30 FOR R = 1 TO 4
 40 PRINT "QUESTION # :"; R
 50 PRINT "1-YES2-NO3-UNDECIDED"
 60 PRINT "WHAT WAS THE RESPONSE : ";
 61 GET C : IF C<1 OR C>3 THEN 61
 65 PRINT C: PRINT
 70 A(R,C) = A(R,C) + 1 : REM UPDATE ELEMENT
 80 NEXT R
 85 PRINT
 90 PRINT "DO YOU WANT TO ENTER ANOTHER" : PRINT "RESPONSE (Y/N) ?";
100 GET A$ : IF A$ = "" THEN 100
110 IF A$ = "Y" THEN 20
120 IF A$ <> "N" THEN 100
130 PRINT "{CLR/HOME}" ; "THE TOTAL RESPONSES WERE:" : PRINT
140 PRINT SPC(18); "RESPONSE"
141 PRINT "QUESTION", "YES", "NO", "UNDECIDED"
142 PRINT "-------- -----------------------------"
150 FOR R = 1 TO 4
160 PRINT R, A(R,1), A(R,2), A(R,3)
170 NEXT R
```

```
RUN

QUESTION # : 1
1-YES  2-NO  3-UNDECIDED
WHAT WAS THE RESPONSE : 1

DO YOU WANT TO ENTER ANOTHER
RESPONSE (Y/N) ?

QUESTION # : 2
1-YES  2-NO  3-UNDECIDED
WHAT WAS THE RESPONSE : 1

And so on...

THE TOTAL RESPONSES WERE:
                         RESPONSE
QUESTION        YES         NO      UNDECIDED
--------        ----------------------------
   1            6           1          0
   2            5           2          0
   3            7           0          0
   4            2           4          1
```

APPENDICI

INTRODUZIONE

Ora che si conosce più a fondo il Commodore 64, è bene dire che il nostro supporto al cliente non si interrompe qui. La cosa potrebbe non essere nota ma la Commodore opera nel settore ormai da oltre 23 anni. Negli anni '70 abbiamo introdotto il primo personal computer (PET). Da allora ci siamo conquistati una posizione di preminenza nel settore computer in molti Paesi del mondo. La nostra capacità di progettare e fabbricare anche i chip, i componenti più importanti e delicati, ci consente di offrire persona computer migliori, avanzati ed a prezzi ben al disotto di quelli che si potrebbe aspettare per questo livello di eccellenza.

La Commodore è impegnata a supportare non soltanto l'utente finale ma anche il rivenditore, le riviste che pubblicano articoli che illustrano nuove applicazioni tecniche e, cosa molto importante, i compilatori di software che producono programmi su cartucce, dischi e nastri da usare con il computer. Noi incoraggiamo a creare dei «club degli utenti Commodore» o ad unirsi a quelli già esistenti per potersi ritrovare tra amici ed imparare nuove tecniche, scambiare idee ed utilizzare in comune le varie scoperte. Pubblichiamo due riviste che contengono suggerimenti di programmazione, informazioni sui nuovi prodotti ed idee per applicazioni del computer (vedere Appendice N).

In Nord America, la Commodore offre il «Commodore Information Network» sul CompuServe Information Service. Per accedere a questa rete, tutto ciò che occorre è un computer Commodore 64 e la nostra cartuccia di interfaccia telefonica a basso costo VICMODEM (o altro modem compatibile).

Le APPENDICI che seguono contengono tabelle, grafici ed altre informazioni che aiutano a programmare il Commodore 64 in modo più veloce e più efficiente. Esse comprendono inoltre informazioni importanti su una grande varietà di prodotti Commodore nonché una bibliografia con oltre venti titoli di libri e riviste che possono essere di aiuto per sviluppare la propria esperienza di programmazione e per mantenersi al corrente sulle informazioni più recenti che riguardano il computer e le periferiche.

APPENDICE A

ACCESSORI E SOFTWARE COMMODORE 64

ACCESSORI

Il Commodore 64 supporterà i dispositivi di memoria e gli accessori Commodore VIC 20 - registratore DATASSETTE, unità disco, modem, stampante - perché il sistema possa espandersi per tenere il passo con le crescenti esigenze dell'utilizzatore.

- Registratore Datassette - Questa unità nastro a basso costo con-sente di memorizzare programmi e dati su nastro in cassetta e riprodurli in un momento successivo. La datassette può anche essere usata per riprodurre programmi pronti.
- Disco - La singola unità disco usa minidischi floppy da 5-1/4", all'incirca le dimensioni di un disco normale da 45 giri, per memorizzare programmi e dati. I dischi consentono un accesso più veloce ai dati e contengono fino a 170000 caratteri di informazione ciascuno. Le unità disco sono “intelligenti" il che significa che dispongono di un proprio microprocessore e di una propria memoria. I dischi non assorbono risorse da Commodore 64, ad esempio usano parte della sua memoria principale.
- Modem - Un dispositivo di comunicazione a basso costo, il VICMODEM consente l'accesso ad altri computer sulle normali linee telefoniche. Grazie ad esso gli utenti avranno accesso alle complete risorse di grandi data base tipo The Source, CompuServe e Dow Jones News Retrieval Service (solo per il Nord America).
- Stampante - La stampante VIC produce tabulati di programmi, da-ti o grafici. Questa stampante ad aghi da 30 caratteri al secondo usa un trascinamoduli per carta normale ed altri materiali di consumo poco costosi. La stampante si collega direttamente al Commodore 64 senza ulteriori interfacce.
- Cartucce di interfaccia - Sarà disponibile per il Commodore 64 un certo numero di cartucce specializzate per consentire il collegamento al sistema di vari dispositivi standard tipi modem, stampanti, unità di controllo e strumenti.

Con la speciale cartuccia IEEE-488, il Commodore 64 potrà supportare l'intera gamma di periferiche CBM comprese le unità disco e le stampanti.
Inoltre, una cartuccia Z80 consentirà di eseguire CP/M su Commodore 64, consentendo l'accesso alla più ampia base di applicazioni per microcomputer attualmente disponibile.

SOFTWARE

Per il Commodore 64 verranno offerte parecchie categorie di software, che consentiranno un'estesa scelta di applicazioni personali, ricreative e didattiche.

Sussidi gestionali

- Un package Electronic Spreadsheet consentirà di pianificare budget ed eseguire le analisi del tipo «cosa succede se?»
- E con il programma optional per grafici, è possibile creare grafici significativi dai dati di tabelle.
- Con il package Financial Planning si potrà facilmente eseguire la pianificazione finanziaria, tipo il calcolo dell'ammortamento di prestiti.
- Un certo numero di programmi Professional Time Management consentirà di gestire gli appuntamenti ed il carico di lavoro.
- Programmi Data Base di facile impiego consentiranno di tener nota delle informazioni... archivi di indirizzi... elenchi telefonici... inventari... ed organizzare le informazioni in forma utile.
- Programmi Word Processing professionali trasformeranno il Commodore 64 in un completo sistema di word processing. La battitura e la revisione di memorandum, lettere ed altro materiale di testo diventeranno facilissime.

Divertimento

- Per il Commodore 64 saranno disponibili su cartucce ad innesto i giochi della migliore qualità che consentiranno ore ed ore di piacevole ed istruttivo divertimento. Questi programmi fanno uso dei grafici ad alta

risoluzione e di tutta la gamma sonora possibile con il Commodore 64.

- Il Commodore 64 consente inoltre di godersi tutto il divertimento e l'eccitazione creati dai giochi MAX in quanto le due macchine usano cartucce completamente compatibili.

Applicazioni didattiche

- Il Commodore 64 è un insegnante che non si stanca mai e che dedica sempre attenzione personalizzata. Oltre alla possibilità di accedere ai numerosi programmi educativi PET, gli ulteriori linguaggi e didattici che verranno disponibili per il Commodore 64 comprendono PILOT, LOGO e altri packages di tipo avanzato.

APPENDICE B

FUNZIONAMENTO AVANZATO DELLA CASSETTA

Oltre a salvare copie di programmi su nastro, il Commodore 64 può anche memorizzare i valori di variabili ed altri elementi di dati in un gruppo detto FILE, consentendo di raccogliere informazioni in numero ancora maggiore di quelle che potrebbero essere contenute in qualsiasi momento nella memoria principale del computer.

Le istruzioni usate con i file di dati sono OPEN, CLOSE, PRINT#, INPUT# e GET#. La variabile di sistema ST (status) viene usata per controllare i delimitatori di nastro.

Nello scrivere i dati sul nastro, vengono usati gli stessi concetti che entrano in gioco quando si visualizzano le informazioni sullo schermo del computer, ma invece di scrivere (PRINI> le informazioni sullo schermo, le informazioni sono scritte (PRINT) su nastro usando una variazione dello stesso comando - PRINT#.

Il programma che segue ne illustra il funzionamento:

```
10 PRINT "WRITE-TO-TAPE-PROGRAM"
20 OPEN 1,1,1, "DATA FILE"
30 PRINT "TYPE DATA TO BE STORED OR TYPE STOP"
50 PRINT
60 INPUT "DATA"; A$
70 PRINT#1, A$
80 IF A$ <> "STOP" THEN 40
90 PRINT
100 PRINT "CLOSING FILE"
110 CLOSE 1
```

La prima cosa che occorre fare è di aprire (OPEN) un file (in questo caso DATA FILE). Se ne occupa la riga 10.

Alla riga 60 il programma richiede i dati che si vogliono salvare su nastro. La riga 70 scrive sul nastro ciò che è stato battuto - cioè contenuto in A$. Ed il processo continua.

Se si vuole interrompere (STOP) la riga 110 chiude (CLOSE) il file.

Per richiamare le informazioni, basta riavvolgere il nastro e provare in questo modo:

```
10 PRINT "READ-TAPE-PROGRAM"
20 OPEN 1,1,0, "DATA FILE"
30 PRINT "FILE OPEN"
40 PRINT
50 INPUT#1, A$
60 PRINT A$
70 IF A$ = "STOP" THEN END
80 GOTO 40
```

Di nuovo, il «DATA FlLE» deve essere per prima cosa aperto (OPEN). Nella riga 50 il programma immette (INPUT) A$ da nastro ed inoltre stampa (PRINT) A$ sullo schermo. Quindi l'intero processo viene ripetuto fino a che non incontra «STOP» il che termina (END) il programma.

Può anche essere usata una variazione di GET - GET# - per leggere i dati da nastro. Sostituire le righe da 50 a 80 nel programma che precede con le seguenti:

```
50 GET#1, A$
60 IF A$  = "" THEN END
70 PRINT A$, ASC(A$)
80 GOTO 50
```

APPENDICE C

BASIC COMMODORE 64

il manuale ha fornito un'introduzione al linguaggio BASIC - quanto basta per avere un'idea della programmazione del computer e di una parte del vocabolario coinvolto. Questa appendice contiene invece un elenco completo delle regole (SINTASSI) del BASIC Commodore 64 unitamente a descrizioni concise. E' bene provare questi comandi. Ricordarsi che non è possibile danneggiare in maniera permanente il computer semplicemente battendo i programmi o manovrando sui tasti che è il modo migliore per imparare.

Questa appendice è divisa in sezioni secondo i diversi tipi di operazioni in BASIC. Queste comprendono:

1. **Variabili e operatori:** Descrive i diversi tipi di variabili, i nomi variabili illeciti e gli operatori aritmetici e logici.
2. **Comandi:** Descrive i comandi usati per lavorare con i programmi, correggere, memorizzare e cancellarli.
3. **Istruzioni:** Descrive le istruzioni di programma BASIC usate nelle righe numerate dei programmi.
4. **Funzioni:** Descrive le funzioni stringa, numeriche e di stampa.

VARIABILI

Il Commodore 64 usa tre tipi di variabili in BASIC, che sono le variabili numeriche reali, le variabili numeriche intere e le variabili stringa(alfanumeriche).

I nomi variabili possono essere costituiti da una singola lettera o da una seguita da un numero o da due lettere.

Una variabile intera viene specificata usando il segno di percento (%) dopo il nome variabile. Le variabili stringa sono seguite del segno del dollaro ($).

Esempi

Nomi variabili reali: A, A5, BZ
Nomi variabili interi: A%, A5%, BZ%
Nomi variabili stringa: A$, A5$, BZ$

Le matrici sono elenchi di variabili con lo stesso nome che usano numeri in più per specificare l'elemento della matrice. Le matrici sono definite usando l'istruzione DIM e possono contenere variabili in virgola mobile, Intere o stringa. Il nome variabile matrice è seguito da una serie di
Parentesi che racchiude il numero delle variabili nella lista.

A(7), BZ%(11), A$(50), PT(20,20)

NOTA: Ci sono tre nomi variabili riservati al Commodore 64 e che non possono essere definiti dall'utente. Queste variabili sono: ST, TI e TI$.

ST è una variabile di status che si riferisce alle operazioni di input/output. Il valore di ST cambia se s'incontra un problema caricando un programma da disco o da nastro.

TI e TI$ sono variabili che riguardano l'orologio in tempo reale incorporato nel Commodore 64. La variabile TI è aggiornata ogni sessantesimo di secondo. Essa inizia a 0 quando il computer viene acceso e viene ripristinata soltanto cambiando il valore di TI$.

TI$ è una stringa che viene costantemente aggiornata dal sistema. I primi due caratteri contengono il numero delle ore, il terzo ed il quarto carattere il numero dei minuti ed il quinto e sesto carattere rappresentano il numero dei secondi. A questa variabile può essere attribuito qualsiasi valore numerico e che sarà quindi aggiornato a

partire da quel momento.

TI$ = 101530» predispone l'orologio a 10:15 e 30 secondi antimeridiane.
Questo orologio viene cancellato quando il computer viene spento e riparte da zero quando il sistema viene di nuovo riacceso.

OPERATORI

Gli operatori aritmetici comprendono i segni seguenti:

+ Addizione
- Sottrazione
* Moltiplicazione
/ Divisione
↑ Elevamento a potenza

Su una riga che contiene più di un operatore c'è un ordine ben definito in cui le operazioni si verificano. Se vengono usate parecchie operazioni insieme sulla stessa riga, il computer assegna la priorità come segue: per primo l'elevamento ad esponente. Successivamente la moltiplicazione e la divisione ed infine l'addizione e la sottrazione.
E' possibile cambiare l'ordine delle operazioni racchiudendo tra parentesi il calcolo da eseguire per primo. Le operazioni racchiuse tra parentesi vengono eseguite prima di tutte le altre.
Ci sono anche delle operazioni di uguaglianza e di disuguaglianza:

= Uguale a
< Minore di
> Maggiore di
<= Minore di o uguale a
>= Maggiore di o uguale a
<> Diverso da

Infine ci sono tre operatori logici:

AND
OR
NOT

Questi vengono usati più spesso per riunire formule multiple nelle
istruzioni IF.. . THEN. Ad esempio:

IF A = B AND C = D THEN 100 (richiede che entrambi le parti siano vere)
IF A = B OR C = D THEN 100 (consente che una o l'altra delle parti sia vera)

COMANDI

CONT (Continua)

Questo comando viene usato per far ripartire l'esecuzione di un programma che e stato interrotto usando il tasto STOP, un'istruzione STOP o un'istruzione END all'interno del programma. Il programma riparte nel punto esatto in cui si era fermato.
CONT non funziona se sono state cambiate o aggiunte righe al programma (o addirittura se viene spostato il cursore) oppure se il programma è interrotto a causa di un errore oppure si è provocato un errore prima di cercare di far ripartire il programma. In questi casi si ottiene un errore del tipo CAN'T CONTINUE (non possono continuare)

LIST

Il comando LIST consente di LISTARE le righe di un programma BASIC
In memoria. E' possibile chiedere di visualizzare l'intero programma o soltanto taluni di una riga.

LIST	Mostra l'intero programma
LIST 10-	Mostra soltanto dalla riga 10 fino alla fine
LIST 10	Mostra soltanto la riga 10
LIST -10	Mostra le righe dall'inizio fino a 10
LIST 10-20	Mostra le righe da 10 a 20 comprese

LOAD

Questo comando viene usato per trasferire in memoria un programma da nastro o da disco in modo da poter usare il programma. Se si batte semplicemente LOAD e si preme RETURN, il primo programma che si trova nella cassetta viene inserito in memoria. Il comando può essere seguito da un nome di programma racchiuso tra virgolette. Il nome può quindi essere seguito da una virgola e da un numero o da una variabile numerica che agisce come numero di dispositivo per indicare la provenienza del programma.

Se non viene indicato alcun numero di dispositivo, il Commodore 64 assume il dispositivo N.1 che è l'unità a cassetta. L'altro dispositivo comunemente usato con il comando LOAD è l'unità disco, che è il dispositivo N.8.

LOAD	Legge il successivo programma sul nastro
LOAD «HELLO»	Ricerca sul nastro il programma denominato HELLO e carica il programma se lo trova
LOAD A$	Cerca un programma il cui nome è nella variabile A$
LOAD «HELLO»,8	Cerca il programma denominato HELLO sull'unità disco
LOAD «*»,8	Cerca il primo programma su disco

NEW

Questo comando cancella l'intero programma in memoria e cancella inoltre qualsiasi variabile che può essere stata usata.

A meno che il programma non sia stato SALVATO (SAVE), va perso. **FARE QUINDI ATTENZIONE QUANDO SI USA QUESTO COMANDO.**

Il comando NEW può anche essere usato come istruzione di programma BASIC. Quando il programma raggiunge questa riga il programma viene cancellato.

Ciò è utile se si vuole lasciare tutto in ordine quando il programma viene eseguito.

RUN

Questo comando provoca l'esecuzione di un programma una volta che tale programma è caricato in memoria. Se RUN non è seguito dal numero di riga, il computer inizia con il numero di riga più basso. Se viene indicato un numero di riga, il programma inizia l'esecuzione alla riga specificata.

RUN   Inizia il programma dal numero di riga più basso
RUN 100   Inizia l'esecuzione alla riga 100
RUN X   errore UNDEFINED STATEMENT (ISTRUZIONE INDEFINITA). Occorre sempre specificare un numero di riga effettivo e non una rappresentazione di variabili

SAVE

Questo comando memorizza il programma correntemente su cassetta o su disco. Se si è soltanto battuto SAVE e RETURN, il programma verrà salvato (SAVE) su cassetta. Il computer non ha alcun modo per sapere se c'è un programma su quel nastro per cui bisogna fare attenzione per non cancellare un programma interessante.

Se si batte SAVE seguito da un nome tra virgolette o da una variabile stringa, Il computer attribuisce quel nome al programma rendendolo più facilmente individuabile e richiamabile in futuro. Il numero può anche essere seguito da un numero di dispositivo.

Dopo il numero di dispositivo, può esserci una virgola e in un secondo numero,0 o 1. Se il secondo numero è

1, il Commodore 64 inserisce un marcatore di FINE NASTRO dopo il programma. Ciò segnalerà al computer di non cercare ulteriormente sul nastro se viene dato un ulteriore comando LOAD. Se si cerca di caricare (LOAD) un programma ed il computer trova uno di questi marcatori, si ottiene un errore FILE NOT FOUND (FILE NON TROVATO).

SAVE	memorizza il programma su nastro senza nome
SAVE «HELLO»	memorizza su nastro con il nome HELLO
SAVE A$	memorizza su nastro con il nome in A$
SAVE «HELLO»,8	memorizza su disco con il nome HELLO
SAVE «HELLO»1,1	memorizza su nastro con il nome HELLO seguito dal marcatore FINE NASTRO

VERIFY

Questo comando fa si che il computer controlli il programma su disco o su nastro a fronte di quello in memoria. Questa è Ia prova che il programma e effettivamente salvato (SAVE) nel caso in cui il nastro o il disco sono difettosi o che qualcosa è andato storto durante l'operazione di SAVE. VERIFY senza null'altro dopo Il comando fa si che il Commodore 64 controlli il successivo programma sul nastro, indipendentemente dal nome, a fronte del programma in memoria.

VERIFY seguito da un nome di programma o da una variabile stringa, cerca quel programma e quindi lo controlla. In questo comando possono anche essere compresi i numeri di dispositivo.

VERIFY	controlla il successivo programma su nastro
VERIFY «HELLO»	cerca HELLO e lo confronta a fronte di quello in memoria
VERIFY «HELLO»,8	cerca HELLO sul disco quindi controlla

CLOSE

Questo comando completa e chiude qualsiasi file usato dall'istruzione OPEN. Il numero the segue CLOSE è Il numero di file da chiudere.

CLOSE 2	viene chiuso soltanto il file N. 2

CLR

Questo comando cancella qualsiasi variabile in memoria ma lascia il programma intatto. Questo comando viene automaticamente eseguito quando viene impartito un comando RUN.

CMD

CMD invia l'output che normalmente andrebbe sullo schermo (e cioè le istruzioni PRINT, LIST ma non POKE sullo schermo) ad un altro dispositivo, che potrebbe una stampante o un file di dati su nastro o su disco. Questo dispositivo o file deve essere per prima cosa aperto (OPEN). Il comando CMD deve essere seguito da un numero o da una variabile numerica che fa riferimento al file.

OPEN 1,4	OPEN(apre) il dispositivo N.4 che è la stampante
CMD 1	Tutto l'output normale va ora alla stampante
LIST	La lista di programma va ora alla stampante e non allo schermo

Per rinviare di nuovo l'output allo schermo, chiudere (CLOSE) il file con CLOSE 1

DATA

Questa istruzione è seguita da un elenco di voci da usare dalle istruzioni READ. Gli elementi possono essere valori numerici o stringhe di testo, separati da virgole. Gli elementi stringa non devono essere all'interno di virgolette a meno che non contengano uno spazio, due punti o virgola. Se due virgole non hanno nulla al loro interno, il valore sarà letto(READ) come zero per un numero o come una stringa vuota.

DATA 12, 14.5, «HELLO, MOM», 3.14, PART 1

DEF FN

Questo comando consente di definire con un breve nome un calcolo
complesso tipo una funzione. Nel caso di una lunga formula usata molte volte nel programma ciò può far risparmiare tempo e spazio.

Il nome della funzione sarà FN e qualsiasi nome variabile lecito (lunghezza 1 o 2 caratteri). Innanzitutto occorre definire la funzione usando L'istruzione DEF seguita dal nome della funzione. Dopo il nome c'è una serie di parentesi che racchiudono una variabile numerica. Segue quindi la formula effettiva che si vuole definire, con la variabile nel punto appropriato. È possibile quindi "richiamare" la formula, sostituendo qualsiasi numero alla variabile.

```
10 DEF FNA(X) = 12*(34.75 - X/.3)
20 PRINT FNA (7)            |
              |             |
              +-------------+ (7 viene inserito nella formula al posto di X)
```

Per questo esempio, il risultato sarà di 137.

DIM (Dimensionamento di una matrice)

Quando si usano più di 11 elementi in una matrice, occorre eseguire un'istruzione DIM per dimensionare la matrice. Tenere presente che l'intera matrice occupa spazio in memoria, per cui è bene non creare matrici molto più lunghe del necessario.

Per rappresentare il numero di variabili create con DIM, moltiplicare il numero totale di elementi in ciascuna dimensione della matrice.

```
10 DIM A(40), B7(15), CC%(4,4,4)
          |      |          |
41 elementi  16 Elementi  125 Elementi
```

È possibile dimensionare più di una matrice in una istruzione DIM.

In ogni caso fare molta attenzione a non dimensionare una matrice più di una volta.

END

Quando incontra un'istruzione END, il programma si interrompe come se fossero esaurite le righe. È possibile usare CONT per far ripartire il programma.

FOR...TO...STEP

Questa istruzione funziona con l'istruzione NEXT per ripetere una sezione di programma un numero di volte prestabilito. Il formato è:

FOR (Nome variabile) = (Inizio del conteggio) TO (Fine del conteggio)
STEP (Variabile di iterazione) = (Conta per)

La variabile di iterazione verrà aggiunta a o sottratta dal programma durante l'esecuzione. Senza la specifica esplicita, STEP si assume pari a 1. Il conteggio di inizio ed il conteggio di fine sono i limiti al valore della variabile di iterazione.

```
10 FOR L = 1 TO 10 STEP.l
20 PRINT L
30 NEXT L
```

La fine del valore di iterazione può essere seguita dalla parola STEP e da un altro numero o variabile. In

questo caso il valore che segue STEP viene aggiunto ogni volta invece di 1. Ciò consente di contare all'indietro o per frazioni.

GET

L'istruzione GET consente di trasferire dati da tastiera, un carattere alla volta. Quando viene eseguito GET, il carattere che viene battuto è assegnato alla variabile. Se non viene battuto alcun carattere, viene assegnato un carattere nullo (vuoto).

GET è seguito da un nome variabile, solitamente una variabile stringa. Se viene usata una variabile numerica e viene premuto un tasto non numerico, il programma si interrompe con un messaggio di errore. L'istruzione GET può essere inserita in un'iterazione, per controllare qualsiasi risultato nullo. Questa iterazione continua fino a che non viene premuto un tasto.

```
10 GET A$: IF A$ = «» THEN 10
```

GET#

L'istruzione GET# è usata con un dispositivo o file precedentemente aperto (OPEN) per immettere un carattere alla volta da quel dispositivo
o file.

```
GET#1,A$
```

Ciò eseguirebbe l'immissione di un carattere da un file di dati.

GOSUB

Questa istruzione è simile a GOTO salvo che il computer ricorda quale riga di programma è stata eseguita per ultima prima di GOSUB. Quando si incontra una riga con un'istruzione RETURN, il programma salta indietro all'istruzione immediatamente seguente il GOSUB. Ciò e utile se c'è una routine che si presenta in parecchie parti del programma. Invece di battere Ia routine ripetutamente, basta eseguire GOSUB ogni volta Che occorre Ia routine stessa.

```
20 GOSUB 800
```

GOTO oppure GO TO

Quando viene raggiunta un'istruzione con il comando GOTO, Ia riga successiva da eseguire sarà quella con il numero di riga che segue la parola GOTO.

IF...THEN

IF. . THEN consente al computer di analizzare una situazione e di intraprendere due azioni possibili in relazione al risultato. Se l'espressione è vera, viene eseguita l'istruzione che segue THEN. Questo può essere qualsiasi istruzione BASIC.

Se l'espressione e falsa Il programma va direttamente alla riga successiva l'espressione che viene valutata può essere una variabile o una formula nel qual caso viene considerata vera se diversa da zero e falsa se zero.

Nella maggior parte dei casi c'è un'espressione che coinvolge operatori relazionali (=, <, >, <=, >=, <>, AND, OR, NOT).

```
10 IF X > 10 THEN END
```

INPUT

L'istruzione INPUT consente al programma di acquisire dati dall'utente assegnando quei dati ad una variabile. Il programma si interrompe, stampa un punto di domanda (?) sullo schermo e attende che l'utente batta la sua risposta e prema RETURN.

INPUT è seguito da un nome variabile o da un elenco di nomi variabili, separati da virgole. Può essere inserito un messaggio tra virgolette prima dell'elenco di nomi variabili da immettere. Se deve essere immessa più di una variabile, le variabili devono essere separate da virgole quando vengono battute.

```
10 INPUT «PLEASE ENTER YOUR FIRST NAME»;A$
20 PRINT «ENTER YOUR CODE NUMBER»; : INPUT B
```

INPUT#

INPUT# è simile a INPUT, ma prende i dati da un file o dispositivo precedentemente aperto (OPEN).

```
10 INPUT#1,A
```

LET

LET non e quasi mai usato in un programma dato che è facoltativo ma l'istruzione è il cuore di tutti i programmi BASIC. Il nome variabile che deve essere assegnato al risultato di un calcolo è sul lato sinistro del segno di uguale e la formula sul lato destro.

```
10 LET A = 5
20 LET D$ = HELLO"
```

NEXT

È sempre usato unitamente all'istruzione FOR. Quando il programma raggiunge un'istruzione NEXT, controlla l'istruzione FOR per accertarsi che il limite dell'iterazione sia stato raggiunto. Se l'iterazione non è finita, la variabile di iterazione viene aumentata del valore da STEP. In caso contrario, l'esecuzione procede con l'istruzione che segue NEXT.

NEXT può essere seguito da un nome variabile, da una lista di nomi variabili separati da virgole. Se non sono indicati nomi, l'ultima iterazione iniziata è quella che viene completata. Se vengono indicate le variabili, queste sono completate nell'ordine da sinistra a destra.

```
10 FOR X = 1 TO 100: NEXT
```

ON

Questo comando attiva I comandi GOTO e GOSUB nelle versioni speciali dell'istruzione IF. ON è seguito da una formula, che viene valutata. Se il risultato del calcolo è 1, viene eseguita la prima riga dell'elenco; se il risultato è 2 viene eseguita a seconda riga e così via. Se il risultato è 0, negativo o maggiore della lista di numeri, la successiva riga eseguita sarà l'istruzione che segue ON.

```
10 INPUT X
20 ON X GOTO 10, 20, 30, 40, 50
```

OPEN

L'istruzione OPEN consente al Commodore 64 di accedere a dispositivi tipo il registratore a cassetta e il disco per i dati, la stampante o anche lo schermo. OPEN è seguito da un numero (da 0 a 255) che è Il numero al quale tutte le istruzioni successive si riferiranno. C'è solitamente un secondo numero dopo Il primo ed è il numero del dispositivo.

I numeri di dispositivo sono:

0 Schermo
1 Cassetta
4 Stampante
8 Disco

Dopo il numero di dispositivo può esserci un terzo numero, separato di nuovo nella virgola, che è l'indirizzo secondario. Nel caso della cassetta questo è 0 per la lettura, 1 per la scrittura e 2 per la scrittura con marcatore di fine nastro.

Nel caso del disco, il numero si riferisce al numero di buffer o di canale.
Nella stampante, l'indirizzo secondario controlla funzioni tipo la stampa ampliata. Vedere Il Manuale di riferimento del Programmatore COMMODORE 64 per ulteriori dettagli.

10 OPEN 1,0	OPEN (APRE) lo SCHERMO come un dispositivo.
20 OPEN 2,1,0,«D»	OPEN (APRE) la cassetta per la lettura; Il file da cercare è D.
30 OPEN 3,4	OPEN (APRE) la stampante.
40 OPEN 4,8,15	OPEN (APRE) il canale di dati sul disco

Vedere inoltre; CLOSE, CMD, GET#, INPUT# e PRINT#, variabili di sistema ST e Appendice B.

POKE

POKE é sempre seguito da due numeri o formule. La prima locazione è una locazione di memoria; il secondo numero è un valore decimale da 0 a 255 che sarà inserito nella locazione di memoria, sostituendovi Il valore precedentemente inserito.

```
10 POKE 53281,0
20 S=4096*13
30 POKE S+29,8
```

PRINT

L'istruzione PRINT è la prima che la maggior parte della gente impara ad usare ma bisogna conoscere le molte variazioni. PRINT può essere seguito da:

stringhe di testo con virgolette
Nomi variabili
Funzioni
Segni di punteggiatura

I segni di punteggiatura sono usati per aiutare a formattare i dati sullo schermo. La virgola divide lo schermo in quattro colonne, mentre il punto e virgola sopprime tutte le spaziature. L'uno o l'altro segno possono essere l'ultimo simbolo su una riga. Ciò fa sì che la successiva cosa stampata (PRINT) agisca come se fosse una continuazione della stessa istruzione PRINT.

```
10 PRINT «HELLO»
20 PRINT «HELLO,»A$
30 PRINT A+B
50 PRINT J;
60 PRINT A,B,C,D
```

Vedere inoltre le funzioni: POS, SPC e TAB

PRINT#

Ci sono poche differenze tra questa istruzione e PRINT. PRINT# è seguita da un numero che fa riferimento al dispositivo o file di dati precedentemente aperto (OPEN). Questo numero è seguito da una virgola e da un elenco da stampare. La virgola ed il punto e virgola hanno lo stesso effetto che hanno in PRINT. Prendere nota che alcuni dispositivi possono non funzionare con TAB e SPC.

```
100 PRINT#1, «DATA VALUES»; A%, B1, C$
```

READ

READ è usato per assegnare informazioni da istruzioni DATA alle variabili cosicché le informazioni possono essere usate, si deve fare attenzione ad evitare di leggere (READ) stringhe dove READ aspetta un numero, Il che darebbe luogo ad un errore TYPE MISMATCH (ERRATO ABBINAMENTO DI TIPO).

REM (Remark - Nota)

REM è una nota indirizzata a chiunque legga una lista del programma. Può spiegare una sezione del programma o fornire ulteriori istruzioni. Le istruzioni REM non influiscono in alcun modo sul funzionamento del programma, salvo che lo allungano. REM può essere seguito da qualsiasi testo.

RESTORE

Quando eseguito in un programma, il puntatore al quale un elemento in un'istruzione DATA verrà letto (READ) viene successivamente ripristinato al primo elemento nella lista. Ciò consente di rileggere le informazioni. RESTORE sta da solo su una riga.

RETURN

Questa istruzione viene sempre usata unitamente a GOSUB. Quando il programma incontra un RETURN, va all'istruzione immediatamente successiva al comando GOSUB. Se non era stato omesso alcun GOSUB si verifica un errore RETURN WITHOUT GOSUB (RETURN SENZA GOSUB)

STOP

Questa istruzione interrompe l'esecuzione del programma. Verrà visualizzato il messaggio BREAK in xxx dove xxx è il numero di riga contenente STOP. il programma può essere fatto ripartire usando il comando CONT. STOP è normalmente usato per correggere programma.

SYS

SYS è seguito da un numero decimale o da un valore numerico nel campo da 0 a 65535. Il programma inizierà quindi l'esecuzione in linguaggio macchina iniziando da quella locazione di memoria. Ciò è simile alla funzione USR ma non consente la trasmissione di parametri.

WAIT

WAIT viene usato per interrompere il programma fino a che i contenuti di una locazione di memoria non cambiano in un modo specifico.
WAIT è seguito da una locazione di memoria (X) e da un massimo di due variabili. Il formato è:

WAIT X,Y,Z

Il contenuto della locazione di memoria viene per prima cosa sottoposto all' operazione di OR-esclusivo con Il terzo numero se presente
E quindi ad un AND-logico con Il secondo numero. Se il risultato è zero,
il programma ritorna a quella locazione di memoria e controlla di nuovo. quando il risultato è diverso da zero, II programma continua con la successiva istruzione.

FUNZIONI NUMERICHE

ABS(X) (valore assoluto)

ABS dà II valore assoluto del numero, senza il segno (- o +). II risultato è sempre positivo.

ATN(X) (arcotangente)

Dà l’angolo, misurato in radianti, la cui tangente è X.

COS(X) (coseno)

Dà il valore del coseno di X, dove X è l’angolo misurato in radianti.

EXP(X)

Dà il valore della costante matematica e (2.71827183) elevata alla potenza di X.

FNxx(X)

Dà il valore della funzione xx definita dall’utente creata in un’istruzione DEF FNxx(X).

INT(X)

Dà il valore troncato di X e cioè senza tutte le cifre decimali alla destra del punto decimale. Il risultato sarà sempre minore di o uguale a X. Pertanto qualsiasi numero negativo con cifre decimali diventerà un intero minore del rispettivo valore corrente.

LOG(X) (logaritmo)

Dà il logaritmo naturale di X. Il logaritmo naturale in base e (vedere EXP(X)). Per convertire nel logaritmo in base 10, basta dividere per LOG(10).

PEEK(X)

Usata per trovare i contenuti della locazione di memoria X, nel campo da 0 a 65535, ottenendo un risultato da 0 e 255. PEEK viene spesso usato unitamente all’istruzione POKE.

RND(X) (numero casuale)

RND(X) dà un numero casuale nel campo da 0 a 1. Il primo numero casuale deve essere generato dalla formula RND(-TI) per iniziare in maniera diversa ogni volta. Dopo ciò, X deve essere un 1 o qualsiasi numero positivo. Se X è zero, il risultato sarà lo stesso numero casuale precedente.
Un valore negativo di X cambia il seme del generatore. L’uso dello stesso numero negativo per X si traduce nella stessa sequenza di numeri «casuali».
La formula per generare un numero compreso tra X e Y è:

```
N = INT (RND(1)*Y)+X)
```

dove:
Y è il limite superiore
X è il campo inferiore dei numeri desiderati

SGN(X) (segno)

Questa funzione dà il segno (positivo, negativo o zero) di X. Il risultato sarà +1 se positivo, 0 se zero e -1 se negativo.

SIN(X) (seno)

SIN(X) è la funzione trigonometrica seno. il risultato sarà il seno di X, dove X è un angolo espresso in radianti.

SQR(X) (radice quadrata)

Questa funzione darà la radice quadrata di X dove X è un numero positivo oppure 0. Se X è negativo, si ottiene un errore ILLEGAL QUANTITY (QUANTITÀ ILLECITA)

TAN(X) (tangente)

Il risultato sarà la tangente di X, dove X è un angolo espresso in radianti.

USR(X)

Quando viene usata questa funzione, Il programma salta ad un programma in linguaggio macchina il cui punto di partenza è contenuto in locazioni di memoria. Il parametro X viene trasmesso al programma in linguaggio macchina che risponderà con un altro valore al programma BASIC. Fare riferimento al Manuale di Riferimento del Programmatore COMMODORE 64, per ulteriori particolari su questa funzione e sulla programmazione in linguaggio macchina.

FUNZIONI STRINGA

ASC(X$)

Questa funzione dà il codice ASCII del primo carattere di X$.

CHR$(X)

Questa è l'opposto di ASC e dà una stringa alfanumerica il cui codice ASCII è X.

LEFT$(X$,X)

Dà una stringa contenente i caratteri X più a sinistra di $X.

LEN(X$)

Dà il numero di caratteri (compresi gli spazi ed altri simboli) nella stringa X$.

MID$(X$,S,X)

Dà una stringa contenente X caratteri iniziando dal carattere Sesimo in X$.

RIGHT$(X$,X)

Dà i caratteri X più a destra in X$.

STR$(X)

Dà una stringa identica alla versione stampata (PRINT) di X.

VAL(X$)

Questa funzione converte X$ in un numero ed è essenzialmente l'operazione inversa di STR$. La stringa viene esaminata partendo dal Carattere più a sinistra verso a quello più a destra per sapere quanti sono i caratteri in formato riconoscibile come numero.

```
10 X = VAL(«123.456»)       X = 123.456
10 X = VAL(«12A13B»)        X = 12
10 X = VAL(«RIU017»)        X = 0
lO X = VAL(«-l.23.45.67»)   X = -1.23
```

ALTRE FUNZIONI

FRE(X)

Questa funzione dà il numero di byte inutilizzati disponibili in memoria, indipendentemente dal valore di X.

POS(X)

Questa funzione dà il numero della colonna (da 0 a 39) alla quale inizierà la successiva istruzione PRINT sullo schermo. X può avere qualsiasi valore e non è usato.

SPC(X)

Viene usata un'istruzione PRINT per saltare X spazi in avanti.

TAB(X)

TAB viene a sua volta usato in un'istruzione PRINT; il successivo elemento da stampare (PRINT) sarà nella colonna X.

APPENDICE D

ABBREVIAZIONI PER LE PAROLE CHIAVE BASIC

Per far risparmiare tempo battendo programmi e comandi, il BASIC COMMODORE 64 consente all'utente di abbreviare la maggior parte delle parole chiave. L'abbreviazione per PRINT è un punto di domanda. Le abberviazioni per altre parole sono ricavate battendo la prima o le prime due lettere della parola, seguite dalla lettera successiva della parola shiftata (preceduta cioè dal tasto SHIFT). Se le abbreviazioni sono usate in una riga di programma, la parola chiave comparirà (LIST) nella forma completa. Notare che alcune delle parole chiave, quando abbreviate, comprendo una parentesi sinistra.

Comando	Abbrevia-zione	Come appare sullo schermo	Comando	Abbrevia-zione	Come appare sullo schermo
ABS	A <SHIFT+B>		CMD	C <SHIFT+M>	
AND	A <SHIFT+N>		PRINT	?	?
ASC	A <SHIFT+S>		PRINT#	P <SHIFT+R>	
ATN	A <SHIFT+T>		READ	R <SHIFT+E>	
CHR$	C <SHIFT+H>		RESTORE	RE <SHIFT+S>	
CLOSE	CL <SHIFT+O>		RETURN	RE <SHIFT+T>	
CLR	C <SHIFT+L>		RIGHT$	R <SHIFT+I>	
CONT	C <SHIFT+O>		RND	R <SHIFT+N>	
DATA	D <SHIFT+A>		RUN	R <SHIFT+U>	
DEF	D <SHIFT+E>		SAVE	S <SHIFT+A>	
DIM	D <SHIFT+I>		SGN	S <SHIFT+G>	
END	E <SHIFT+N>		SIN	S <SHIFT+I>	
EXP	E <SHIFT+X>		SPC(	S <SHIFT+P>	
FOR	F <SHIFT+O>		SQR	S <SHIFT+Q>	
FRE	F <SHIFT+R>		STATUS	ST	ST
GET	G <SHIFT+E>		STEP	ST <SHIFT+E>	
GOSUB	GO <SHIFT+S>		STOP	S <SHIFT+T>	
GOTO	G <SHIFT+O>		STR$	ST <SHIFT+R>	
INPUT#	I <SHIFT+N>		SYS	S <SHIFT+Y>	
LET	L <SHIFT+E>		TAB	T <SHIFT+A>	
LEFT$	LE <SHIFT+F>		THEN	T <SHIFT+H>	
LIST	L <SHIFT+I>		SAVE	S <SHIFT+A>	

LOAD	L <SHIFT+O>		TIME	TI	TI
MID$	M <SHIFT+I>		TIME$	TI$	TI$
NEXT	N <SHIFT+E>		USR	U <SHIFT+S>	
NOT	N <SHIFT+O>		VAL	V <SHIFT+A>	
OPEN	O <SHIFT+P>		VERIFY	V <SHIFT+E>	
PEEK	P <SHIFT+E>		WAIT	W <SHIFT+A>	
POKE	P <SHIFT+O>				

APPENDICE E

CODICI DELLO SCHERMO VIDEO

La tabella che segue elenca tutti i caratteri della serie COMMODORE 64. Essa mostra quale numero deve essere inserito (POKE) nella memoria dello schermo (locazioni da 1024 a 2023) per ottenere un carattere desiderato. Viene inoltre indicato quale carattere corrisponde ad un numero osservato dallo schermo (PEEK).

Sono disponibili due serie di caratteri ma solo una alla volta. Cioè significa che non è possibile avere sullo schermo contemporaneamente caratteri di due serie. Si passa dall'una all'altra serie tendendo abbassati simultaneamente i tasti SHIFT e COMMDODORE.

Dal BASIC, POKE 53272,29 passerà nel modo maiuscolo e POKE 53272,31 passerà nel modo minuscolo.

Qualsiasi numero sulla tabella può anche essere visualizzato in NEGATIVO. Il codice del carattere negativo può essere visualizzato in NEGATIVO. Il codice del carattere negativo può essere ottenuto aggiungendo 128 ai valori indicati.

Se si vuole visualizzare un cerchio pieno nella locazione 1504, inserire (POKE) il codice per il cerchio (81) nella locazione 1504: POKE1504,81.

C'è una corrispondente locazione di memoria per controllare il colore di ciascun carattere visualizzato sullo schermo (locazioni da 55296 a 56295). Per cambiare il colore del cerchio in giallo (codice colore 7), occorrerà inserire (POKE) la corrispondente locazione di memoria (55776) con il colore del carattere: POKE55776,7.

Fare riferimento all'Appendice G per le mappe complete della memoria dei colori e dello schermo unitamente al rispettivo codici dei colori.

CODICI DELLO SCHERMO

SET 1	SET 2	POKE	SET 1	SET 2	POKE	SET 1	SET 2	POKE
@		0	+		43		V	86
A	a	1	,		44		W	87
B	b	2	-		45		X	88
C	c	3	.		46		Y	89
D	d	4	/		47		Z	90
E	e	5	0		48			91
F	f	6	1		49			92
G	g	7	2		50			93
H	h	8	3		51			94
I	i	9	4		52			95
J	j	10	5		53	SPACE		96
K	k	11	6		54			97
L	l	12	7		55			98
M	m	13	8		56			99
N	n	14	9		57			100
O	o	15	:		58			101
P	p	16	;		59			102
Q	q	17	<		60			103
R	r	18	=		61			104
S	s	19	>		62			105
T	t	20	?		63			106
U	u	21			64			107
V	v	22		A	65			108

W	w	23	B		66			109
X	x	24	C		67			110
Y	y	25	D		68			111
Z	z	26	E		69			112
[		27	F		70			113
$		28	G		71			114
]		29	H		72			115
^		30	I		73			116
<-		31	J		74			117
SPACE		32	K		75			118
!		33	L		76			119
"		34	M		77			120
#		35	N		78			121
$		36	O		79			122
%		37	P		80			123
&		38	Q		81			124
'		39	R		82			125
(		40	S		83			126
)		41	T		84			127
*		42	U		85			

I codici da 128 a 255 sono le immagini in negativo dei codici da 0 a 127

APPENDICE F

CODICI ASCII E CHR$

Questa Appendice mostra quali caratteri compariranno se si esegue PRINT CHR$(X), per tutti i possibili valori di X. Essa indicherà anche i valori ottenuti battendo PRINT ASCII ("X") dove x è qualsiasi carattere che è possibile battere. Ciò è utile per valutare il carattere ricevuto un istruzione GET, convertendo maiuscole/minuscole e stampando i comandi basati sui caratteri (come ad esempio il passaggio maiuscolo/minuscolo) che non possono essere racchiusi tra virgolette.

STAMPA	CHR$	STAMPA	CHR$	STAMPA	CHR$	STAMPA	CHR$
	0	0	48		96	{blk}	144
	1	1	49		97	{up}	145
	2	2	50		98	{rvs off}	146
	3	3	51		99	{clr home}	147
	4	4	52		100	{inst del}	148
{wht}	5	5	53		101	{brown}	149
	6	6	54		102	{lt. red}	150
	7	7	55		103	{grey 1}	151
disSHIFT+C=	8	8	56		104	{grey 2}	152
abiSHIFT+C=	9	9	57		105	{lt.green}	153
	10	:	58		106	{lt.blue}	154
	11	;	59		107	{grey 3}	155
	12	<	60		108	{purple}	156
RETURN	13	=	61		109	{left}	157
MINUSCOLO	14	>	62		110	{yellow}	158
	15	?	63		111	{cyan}	159
	16	@	64		112	SPACE	160
{down}	17	A	65		113		161
{rvs on}	18	B	66		114		162
{clr home}	19	C	67		115		163
{inst del}	20	D	68		116		164

	21	E	69		117		165
	22	F	70		118		166
	23	G	71		119		167
	24	H	72		120		168
	25	I	73		121		169
	26	J	74		122		170
	27	K	75		123		171
{red}	28	L	76		124		172
{right}	29	M	77		125		173
{grn}	30	N	78		126		174
{blu}	31	O	79		127		175
SPACE	32	P	80		128		176
!	33	Q	81	{orange}	129		177
"	34	R	82		130		178
#	35	S	83		131		179
$	36	T	84		132		180
%	37	U	85	f1	133		181
&	38	V	86	f3	134		182
'	39	W	87	f5	135		183
(	40	X	88	f7	136		184
)	41	Y	89	f2	137		185
*	42	Z	90	f4	138		186
+	43	[	91	f6	139		187
,	44	pound	92	f8	140		188
-	45	]	93	shift+ret.	141		189
.	46	^	94	maiuscolo	142		190
/	47	{arrow left}	95		143		191

Codici da 192 a 223 identici ai codici da 96 a 127
Codici da 224 a 254 identici ai codici da 160 a 190
Codice 255 identico a 126

APPENDICE G

MAPPA DI MEMORIA DEI COLORI E DELLO SCHERMO

Le tabelle che seguono elencano quali sono le locazioni di memoria che controllano il posizionamento dei caratteri sullo schermo e le locazioni usate per cambiare i singoli colori dei caratteri. Mostrano inoltre i codici colore dei caratteri.

MAPPA DI MEMORIA DELLO SCHERMO

```
                            COLONNE                              1063
      0             10             20             30            39 /
     +------------------------------------------------------------/
1024 |                                                            |  0
1064 |                                                            |
1104 |                                                            |
1144 |                                                            |
1184 |                                                            |
1224 |                                                            |
1264 |                                                            |
1304 |                                                            |
1344 |                                                            |
1384 |                                                            |
1424 |                                                            | 10
1464 |                                                            |
```

I valori effettivi da inserire (POKE) in una locazione di memoria dei colori per cambiare un colore di un carattere, sono:

0	NERO	8	ARANCIONE
1	BIANCO	9	MARRONE
2	ROSSO	10	ROSSO Chiaro
3	BLU VERDE	11	GRIGIO 1
4	PORPORA	12	GRIGIO 2
5	VERDE	13	VERDE Chiaro
6	BLU	14	BLU Chiaro
7	GIALLO	15	GRIGIO 3

Per esempio per cambiare in rosso il colore di un carattere disposto nell'angolo superiore sinistro dello schermo, battere: POKE66296,2.

MAPPA DI MEMORIA DEI COLORI

```
                          COLONNE                     55335
   0            10            20            30          39 /
  +--------------------------------------------------------/
55296|                                                        |  0
55336|                                                        |
55376|                                                        |
55416|                                                        |
55456|                                                        |
55496|                                                        |
55536|                                                        |
55576|                                                        |
55616|                                                        |
55656|                                                        |
55696|                                                        | 10
55736|                                                        |
55776|                                                        |   ROW
55816|                                                        |
55856|                                                        |
55896|                                                        |
55936|                                                        |
55976|                                                        |
56016|                                                        |
56056|                                                        |
56096|                                                        | 20
```

```
56136|                                                             |
56176|                                                             |
56216|                                                             |
56256|                                                             | 24
     +-------------------------------------------------------------\
                                                                    56295
```

APPENDICE H

DERIVAZIONI DI FUNZIONI MATEMATICHE

Le funzioni che non sono incorporate nel BASIC del COMMODORE 64, possono essere calcolate come segue:

FUNZIONE	EOUIVALENTE VIC BASIC
SECANTE	SEC(X)= 1/COS(X)
COSECANTE	CSC(X) 1/SIN(X)
COTANGENTE	COT(X)= 1/TAN(X)
SENO INVERSO	ARCSIN(X)=ATN(X/SQR(-X*X+1))
COSENO INVERSO	ARCCOS(X)=-ATN(X/SQR(-X*X+1))+{pi}/2
SECANTE INVERSA	ARCSEC(X)=ATN(X/SQR(X*X-1))
COSECANTE INVERSA	ARCCSC(X)=ATN(X/SQR(X*X-1))+(SGN(X)-1*{pi}/2
COTANGENTE INVERSA	ARCOT(X)=ATN(X)+{pi}/2
SENO IPERBOLICO	SINH(X)=(EXP(X)-EXP(-X))/2
COSENO IPERBOLICO	COSH(X)=(EXP(X)+EXP(-X))/2
TANGENTE IRERBOLICA	TANH(X)=EXP(-X)/(EXP(X)+EXP(-X))*2+1
SECANTE IPERBOLICA	SECH(X)=2/(EXP(X)+EXP(-X))
COSECANTE IPERBOLICA	CSCH(X)=2/(EXP(X)-EXP(-X))
COTANGENTE IPERBOLICA	COTH(X)=EXP(-X)/(EXP(X)-EXP(-X))*2+1
SENO IPERBOLICO INVERSO	ARCSINH(X)=LOG(X+SQR(X*X+1))
COSENO IPERBOLICO INVERSO	ARCCOSH(X)=LOG(X+SQR(X*X-1))
TANGENTE IPERBOLICA INVERSA	ARCTANH(X)=LOG((1+X)/(1-X))/2
SECANTE IPERBOLICA INVERSA	ARCSECH(X)=LOG((SQR(-X*X+1)+1/X)
COSECANTE IPERBOLICA INVERSA	ARCCSCH(X)=LOG((SGN(X)*SQR(X*X+1/X)
COTANGENTE IPERBOLICA INVERSA	ARCCOTH(X)=LOG((X+1)/(X-1))/2

APPENDICE I

CONFIGURAZIONE DEI PIN PER I DISPOSITIVI DI INPUT/OUTPUT

Questa appendice si propone di mostrare quali collegamenti possono essere realizzati sul COMMODORE 64.

1) I/O dei giochi
2) Fessura per cartuccia
3) Audio/video
4) I/O seriale (disco/stampante)
5) Output del modulatore
6) Cassetta
7) Connettore per utente

Connettore di controllo 1

Pin	Tipo	Nota
1	JOYA0	
2	JOYA1	
3	JOYA2	
4	JOYA3	
5	POT AY	
6	BUTTON A/LP	
7	+5V	MAX 100 mA
8	GND	
9	POT AX	

Connettore di controllo 2

Pin	Tipo	Nota
1	JOYB0	
2	JOYB1	
3	JOYB2	
4	JOYB3	
5	POT BY	
6	BUTTON B	
7	+5V	MAX 100 mA
8	GND	
9	POT BX	

[IMMAGINE PAG 141]

Fessura di espansione per cartuccia

Pin	Tipo
22	GND
21	CD0
20	CD1
19	CD2
18	CD3
17	CD4
16	CD5
15	CD6
14	CD7
13	DMA
12	BA
11	ROML
10	1/02
9	EXROM
8	GAME
7	1/01
6	Dot Clock
5	CR/W
4	IRQ
3	+5V
2	+5V
1	GND
Z	GND
Y	CA0
X	CA1
W	CA2
V	CA3
U	CA4
T	CA5
S	CA6
R	CA7
P	CA8
N	CA9
M	CA10
L	CA11
K	CA12
J	CA13
H	CA14
F	CA15
E	S02
D	NMI
C	RESET
B	ROMH
A	GND

[IMMAGINE PAG 142]

Audio/Video

Pin	Tipo	Nota
1	LUMINANZA	
2	MASSA	
3	USCITA AUDIO	
4	USCITA VIDEO	
5	INGRESSO AUDIO	

[IMMAGINE PAG 142]

I/O seriale

Pin	Tipo
1	SRQIN SERIALE
2	MASSA
3	IN/OUT ATN SERIALE
4	IN/OUT CLK SERIALE
5	IN/OUT DATI SERIALE
6	RESET

[IMMAGINE PAG 142]

Cassetta

Pin	Tipo
A-1	MASSA
B-2	+5V
C-3	MOTORE PER CASSETTA
D-4	LETTURA CASSETTA
E-5	SCRITTURA CASSETTA
F-6	RILEVAMENTO CASSETTA

[IMMAGINE PAG 143]

I/O Utente

Pin	Tipo	Nota
1	GND	
2	+5V	MAX 100 mA
3	RESET	
4	CNT1	
5	SP1	
6	CNT2	
7	SP2	
8	PC2	
9	SER. ATN IN	
10	9 VAC	MAX 100 mA
11	9 VAC	MAX 100 mA
12	GND	
A	GND	
B	FLAG2	
C	PB0	
D	PB1	
E	PB2	
F	PB3	
H	PB4	
J	PB5	
K	PB6	
L	PB7	
M	PA2	
N	GND	

[IMMAGINE PAG 143]

APPENDICE J

PROGRAMMI DA PROVARE

È stato inserito un certo numero di programmi che è possibile provare con il COMMODORE 64. Questi programmi risulteranno sia divertenti che utili.

```
100 print"{reverse on} indovinello   "
120 input "{reverse on} vuoi istruzzioni?";z$:ifasc(z$)=78goto250
130 print"{reverse on} prova ad indovinare le parole misteriose formate da 5 lettere"
140 print"{reverse on} soltantto parole di 5 lettere sono riconosciute"
150 print"{reverse on} anche nomi...."
160 print"{reverse on} ti saranno dette"
170 print"{reverse on} le lettere sbagliate"
180 print"{reverse on} haa: il trtrucco varia"
190 print"{reverse on} da un indovinello all'altro;cosiche "
200 print"{reverse on} se indovini due lettere "
210 print"{reverse on} hai una idea di quelle che sono"
220 print"{reverse on} le lettere mancanti...."
250 data ubtup,tfejb,dbsub,cvtub,cbsdb
260 data vtdjp,qptub,mjcsp,tvpmb,svpmp
270 data gjfsf,svpub,cjssb,tdbmb,tqjob
280 data wjhob,ipccz,csjob,uspub,gvohp
290 data qftdb,hjpjb,qjbop,gsfop,csvob
300 data tdvep,uvsop,tqbeb,csbop,mfqsf
310 data hsbop,ufoeb,hvbep,nbsdb,wfoup
320 data qbmmb,sfuub,hsbep,hpmjb,npouf
```

```
330 data bdrvb,sbejp,dbqsb,usfop,spttp
340 data djfmp,nbsub,wfsef,cpddb,tqptb
400 n=50
410 dimn$(n),z(5),y(5)
420 forj=1ton:readn$(j):nextj
430 t=ti
440 t=t/1000:ift>=1thengoto440
450 z=rnd(-t)
500 g=0:n$=n$(rnd(1)*n+1)
510 print"{reverse on} ho una parola composta da 5 lattere:":ifr>0goto560
520 print"indovina(con la esatta quantita'di lettere"
530 print"e ti diro' quante lettere giuste"
550 print"hai indovinato"
560 g=g+1:input"nella tua parola";z$
570 iflen(z$)<>5thenprint"devi indovinare parole di sole 5 lettere":goto560
580 v=0:h=0:m=0
590 forj=1to5
600 z=asc(mid$(z$,j,1)):y=asc(mid$(n$,j,1))-1:ify=64theny=90
610 ifz=65orz=69orz=73orz=79orz=85orz=89thenv=v+1
630 ifz=ythenm=m+1
640 z(j)=z:y(j)=y:nextj
650 ifm=5goto800
660 ifv=0orv=5thenprint"suvvia...che tipo di parola e' questa?":goto560
670 forj=1to5:y=y(j)
680 fork=1to5:ify=z(k)thenh=h+1:z(k)=0:goto700
690 nextk
700 nextj
710 print"{right*20}";h;"lettere"
720 ifg<30goto560
730 print"e' meglio che ti dica ...la parola era..";
740 forj=1to5:printchr$(y(j));:nextj
750 print"'":goto810
800 print"indovinato!! con sole "g" prove"
810 input"{reverse on} un altra parola";z$
820 r=1:ifasc(z$)<>78goto500

1 rem***sequenza
50 dima$(26)
100 z$="abcdefghijklmnopqrstuvwxyz"
110 z1$="12345678901234567890123456"
200 print"{down*2} immettere la lunghezza della stringa in sequenza{down}"
220 input"la massima lunghezza e'  26";s%
230 ifs%<1ors%>26 then200
240 s=s%
300 for i=1tos
310 a$(i)=mid$(z$,i,1)
320 nexti
400 rem stringa random
420 fori=1tos
430 k=int(rnd(1)*s+1)
440 t$=a$(i)
450 a$(i)=a$(k)
460 a$(k)=t$
470 nexti
480 gosub950
595 t=0
600 rem substringa in reverse
605 t=t+1
```

```
610 input"quanti in reverse  ";r%
620 if r%=0goto900
630 if r%>0andr%<=sgoto650
640 print" deve essere tra 1 e ";s:goto610
650 r=int(r%/2)
660 fori=1tor
670 t$=a$(i)
680 a$(i)=a$(r%-i+1)
690 a$(r%-i+1)=t$
700 nexti
750 gosub950
800 c=1:fori=2tos
810 ifa$(i)>a$(i-1)goto830
820 c=0
830 nexti
840 ifc=0goto600
850 print"{down} ci sei riuscito in ";t;"volte"
900 rem controlla per un altro gioco
910 input"{down} vuoi giocare ancora ";y$
920 ifleft$(y$,1)="s"goto200
930 end
950 print
960 printleft$(z1$,s)
970 fori=1tos:printa$(i);:nexti
980 print"{down}"
990 return
```

```
50 rem pianoforte elettronico
100 print"{clear} {reverse on} {right} {right} {221} {right} {right} {right} {221}
{right} {right} {221} {right} {right} "
110 print" {reverse on} {right} {right} {221} {right} {right} {right} {221} {right}
{right} {221} {right} {right} "
120 print" {reverse on} {right} {right} {221} {right} {right} {right} {221} {right}
{right} {221} {right} {right} "
130 print"{160}{reverse on} {221} {221} {221} {221} {221} {221} {221} {221} {221}
{221} {221} {221} "
140 print" {reverse
on}q{221}w{221}e{221}r{221}t{221}y{221}u{221}i{221}o{221}p{221}@{221}*{221}^{221};{
221}={221}"
150 print"{down}'spazio'per assolo o accompagnamento"
160 print"{down}'f1,f3,f5,f7' selezione delle ottave"
170 print"{down}'f2,f4,f6,f8'forma d'onda{down}"
180 print"aspetta, bisogna settare la frequenza..."
190 si=13*4096+1024:dimf(26):dimk(255)
200 fori=0to28:pokesi+i,0:next
210 f1=7939:fori=1to26:f(27-i)=f1*5.8+30:f1=f1/2^(1/12):next
220 k$="1q2w3er5t6y7ui9o0p@-*^{home};="
230 fori=1tolen(k$):k(asc(mid$(k$,i)))=i:next
240 print"{up*2}{space*40}"
250 an=0:ab=0:ha=15:au=9:hh=ha*16+au:as=an*16+ab:wf=16:m=1:ok=4:hb=256:z=0
260 fori=0to2:pokesi+5+i*7,as:pokesi+6+i*7,ht
270 pokesi+2+i*7,4000and255:pokesi+3+i*7,4000/256:next
280 pokesi+24,15+16+64:pokesi+23,7:rem cancella filtro
300 geta$:ifa$=""then300
310 fr=f(k(asc(a$)))/m:fl=si+v*7:w=fl+4:iffr=zthen500
320 pokefl+6,z:rem dec/sus terminato
325 pokefl+5,z:rem att/rel terminato
330 pokew,8:pokew,0:rem reset
```

```
340 pokefl,fr-hb*int(fr/hb):rem set lo
350 pokefl+1,fr/hb:rem set hi
360 pokefl+6,hh:rem set dec/sus
370 pokew,wf+1:fori=1to50*an:next
375 pokew,wf:rem impulso
380 ifp=1thenv=v+1:ifv=3thenv=0
400 goto300
410 rem"tasti funzione
500 ifa$="{f1}"thenm=1:ok=4:goto300
510 ifa$="{f3}"thenm=2:ok=3:goto300
520 ifa$="{f5}"thenm=4:ok=2:goto300
530 ifa$="{f7}"thenm=8:ok=1:goto300
540 ifa$="{f2}"thenwf=16:goto300
550 ifa$="{f4}"thenwf=32:goto300
560 ifa$="{f6}"thenwf=64:goto300
570 ifa$="{f8}"thenwf=128:goto300
580 ifa$=" "thenp=1-p:hh=(hhandnot2)or(nothhand2):goto300
590 ifa$="{clear}"then200
600 goto300
```

APPENDICE K

CONVERSIONE DI PROGRAMMI BASIC STANDARD IN BASIC COMMODORE 64

Se si dispone di programmi scritti in un BASIC diverso da quello del COMMODORE 64, potranno essere necessari piccoli aggiustamenti prima di poterli eseguire sul COMMODORE 64. Ecco alcuni suggerimenti per rendere più facile la conversione.

Dimensioni di stringa

Cancellare tutte le istruzioni che vengono usate per dichiarare la lunghezza di stringhe. Un'istruzione tipo DIM A$(I,J) che dimensiona una matrice stringa per J elementi di lunghezza I, deve essere convertita nell'istruzione BASIC COMMODORE DIM A$(J).

Alcuni BASIC usano una virgola o una e commerciale (&) per la concatenazione di stringa. Ciascuna di queste deve essere cambiata in un segno più, che è l'operatore BASIC COMMODORE per la concatenazione di stringhe.

Nel BASIC COMMODORE 64, sono usate le funzioni MID$, RIGHT$ e LEFT$ per estrarre substringhe di stringhe. Forme tipo A$(I) per accedere all'i-esimo carattere in A$ o A$(I,J) per prelevare una substringa di A$ dalla posizione I alla J, devono essere cambiate come segue:

Altri BASIC	BASIC 64 COMMODORE
A$(I) = X$	A$ = LEFT$(A$,I-1)+X$+MID$(A$,I+1)
A$(I,J) = X$	A$ = LEFT$(A$,I-1)+X$+MID$(A$,J+1)

Assegnazioni multiple

Per porre B e C uguale a zero, alcuni BASIC consentono istruzioni della forma:

```
10 LET B=C=0
```

Il BASIC COMMODORE 64 interpreterebbe il secondo segno di uguale come operatore logico e fisserebbe B =-1 se C=0. Per contro, occorre convertire questa istruzione nella seguente:

```
10 C=0 : B=0
```

Istruzioni multiple

Alcuni BASIC usano una barra rovesciata (\) per separare più istruzioni su una riga. Con il BASIC COMMODORE 64, occorre separare tutte le istruzioni mediante un due punti (:).

Funzioni MATEMATICHE

i programmi che usano le funzioni Matematiche disponibili su alcuni BASIC devono essere riscritti usando le iterazioni FOR...NEXT per poterli eseguire correttamente.

APPENDICE L

MESSAGGI DI ERRORE

Questa Appendice contiene un elenco completo dei messaggi di errore generati dal COMMODORE 64, con una descrizione delle relative cause.

BAD DATA (Dati errati) La stringa di dati è stata ricevuta da un file aperto ma il programma aspettava dati numerici.

BAD SUBSCRIPT (Indice errato) Il programma stava cercando di tare riferimento ad un elemento di una matrice il cui numero è al di fuori del campo specificato nell'istruzione DIM.

CAN'T CONTINUE (Non posso continuare) Il comando CONT non funziona sia perché il programma non è mai stato eseguito (RUN) oppure perché c'è stato un errore oppure perché è stata corretta una riga.

DEVICE NOT PRESENT (Dispositivo non presente) Il richiesto dispositivo di I/O non era disponibile per un'istruzione OPEN, CLOSE, CMD, PRINT#, INPUT# o GET#

DIVISION BY ZERO (Divisione per zero) La divisione per zero è una incongruenza matematica e non è ammessa.

EXTRA IGNORED (Extra ignorati) Sono stati battuti troppi elementi di dati in risposta ad un'istruzione INPUT. Sono stati accettati soltanto i primi elementi.

FILE NOT FOUND (File non trovato) Se si stava cercando un file su nastro e si è trovato un marcatore di fine nastro. Se si cercava su disco, non esiste file con quei nome.

FILE NOT OPEN (File non aperto) Il file specificato in un'istruzione CLOSE, CMD, PRINT#, INPUT# o GET# deve essere per prima cosa aperto (OPEN).

FILE OPEN (File aperto) È stato compiuto un tentativo di aprire un file già aperto.

FORMULA TOO COMPLEX (Formula troppo complessa) L'espressione stringa che viene calcolata deve essere suddivisa in almeno due parti perché il sistema la possa elaborare.

ILLEGAL DIREKT (Illecito nel modo diretto) L'istruzione INPUT può essere usata soltanto nell'ambito di un programma e non nel modo diretto.

ILLEGAL OUANTITY (Quantità illecita) Un numero usato come argomento di una funzione o istruzione è fuori dal campo ammesso. LOAD (Caricamento) C'è un problema con il programma sul nastro.

NEXT WITHOUT FOR (NEXT senza FOR) Ciò è provocato dalla nidificazione di iterazioni scorrette o dalla presenza di un nome variabile in un'istruzione NEXT che non corrisponde a quella nell'istruzione FOR.

NOT INPUT FILE (Non è un file di input) È stato compiuto un tentativo di INPUT o di GET su un file che era stato specificato solo per l'output.

NOT OUTPUT FILE (Non è un file di output) È stato compiuto un tentativo di PRINT dati su un file che era stato specificato solo per l'input.

OUT OF DATA (non più dati) È stata eseguita un'istruzione READ ma non c'erano più dati da leggere (READ) in un'istruzione DATA.

OUT OF MEMORY (Mancanza di memoria) Non c'è altra RAM disponibile per il programma o le variabili. Ciò può anche verificarsi quando sono stati nidificate troppe iterazioni FOR o quando ci sono in atto troppi GOSUB.

OVERFLOW (Superamento di capacità) Il risultato di un calcolo è maggiore del numero massimo ammesso che è i .70141884E+38.

REDIM'D ARRAY (Matrici ridimensionate) Una matrice può essere dimensionata (DI M) solo una volta. Se viene usata una variabile matrice prima che quella matrice sia dimensionata (DIM), viene eseguita un'operazione DIM automatica su quella matrice portando il numero di elementi a 10 e qualsiasi successiva DIM provoca questo errore.

REDO FROM START (Ripetere dall'inizio) Sono stati battuti dati di alfabetici durante un'istruzione INPUT quando si aspettavano dati numerici. Basta ribattere l'entrata corretta e il programma continua da solo.

RETURN WITHOUT GOSUB (RETURN senza GOSUB) Si è incontrata un'istruzione RETURN e non è stato

emesso un comando GOSUB.

STRING TOO LONG (Stringa troppo lunga) Una stringa può contenere fino a 255 caratteri.

?SYNTAX ERROR (Errore di sintassi) Un'istruzione non riconosciuta dal COMMODORE 64. (Parentesi mancanti o in più, parole chiave scritte erroneamente, ecc.).

TYPE MISMATCH (Errato abbinamento di tipo) Questo errore si verifica quando viene usato un numero in luogo di una stringa o viceversa.

UNDEF'D FUNCIION (Funzione non definita) È stato fatto riferimento ad una funzione definita dall'utente che non è mai stata definita usando l'istruzione DEF FN.

UNDEF'D STATEMENT (Istruzione non definita) È stato compiuto un tentativo di GOTO o GOSUB o di RUN relativamente a un numero di riga che non esiste.

VERIFY (Verifica) Il programma su nastro o su disco non corrisponde al programma correntemente in memoria.

APPENDICE M

BIBLIOGRAFIA

Addison Wesley — **BASIC and the Personal Computer,**
Dwyer and Critchfiel

Dilithium Press — **BASIC Basic-English Dictionary tor the Pet,**
Larry Noonan

Faulk Baker Associates — **MOS Programming Manual,**
MOS Technology

Hayden Book Co. — **BASIC from the Ground Up,**
David E. Simon «I Speak Basic»

Littie, Brown and Co. — **Computer Games for Businesses, Schools and Homes, J. Victor**
Nagigian and William S. Hodges

The Computer Tutor: Learning Activities for Homes and Schools,
Gary W. Orwing, University of Central Florida and William S. Hodges

McGraw Hill — **Hands-On BASIC with a Pet,**
Herbert D. Peckman

Osborne/McGraw Hill — **Pet/CBM Personal Computer Guide,**
Carrol S. Donahue

Osborne CP/M User «Guide»,
Thom Hogan

Same Common Basic Programs,
Lon Poole and Mary Borchers

The 8086 Book,
Russel Rector and George Alexy

Reston Publishing Co. — **Pet and the IEEE 488 Bus (GPIB),**
Eugene Fischer and C. W. Jensen

PET BASIC,
Ramon Zamora, William Scarvie and Bob Albrecht

Pet Games and Recreation,

Mac Ogelsby, Len Lindsey and Dorothy Kunkin

Commodore Magazines fornisce le informazioni più aggiornate per il COMMODORE 64. Due delle più diffuse pubblicazioni alle quali è utile abbonarsi sono:

COMMODORE - The Microcomputer Magazine viene pubblicato mensilmente ed è disponibile in abbonamento ($ 15.00 per 6 mesi in U.S.A. e $ 25.00 per 6 mesi in tutto il mondo).

POWER PLAY - Computing Magazine è pubblicato trimestralmente ed è disponibile in abbonamento ($ 10.00 all'anno in U.S.A. e $ 15.00 all'anno in tutto in mondo).

APPENDICE N

ORDINAMENTO REGISTRI SPRITE

Indirizzo Basic VIC = 53248 (Dec) = D000 (Esadec)

Registri# Dec	Esa	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	
0	0	S0X7							S0X0	SPRITE 0 X Component
1	1	S0Y7							S0Y0	SPRITE 0 Y Component
2	2	S1X7							S1X0	SPRITE 1 X
3	3	S1Y7							S1Y0	SPRITE 1 Y
4	4	S2X7							S2X0	SPRITE 2 X
5	5	S2Y7							S2Y0	SPRITE 2 Y
6	6	S3X7							S3X0	SPRITE 3 X
7	7	S3Y7							S3Y0	SPRITE 3 Y
8	8	S4X7							S4X0	SPRITE 4 X
9	9	S4Y7							S4Y0	SPRITE 4 Y
10	A	S5X7							S5X0	SPRITE 5 X
11	B	S5Y7							S5Y0	SPRITE 5 Y
12	C	S6X7							S6X0	SPRITE 6 X
13	D	S6Y7							S6Y0	SPRITE 6 Y
14	E	S7X7							S7X0	SPRITE 7 X
15	F	S7Y7							S7Y0	SPRITE 7 Y
16	10	S7X8	S6X8	S5X8	S4X8	S3X8	S2X8	S1X8	S0X8	Bit alti del valore - X

17	11	RC8	ECM	BMM	BLNK	RSEL	YSCL2	YSCL1	YSCL0	
18	12	RC7	RC6	RC5	RC4	RC3	RC2	RC1	RC0	TAVOLA
19	13	LPX7							LPX0	LIGHT PEN X
20	14	LPY7							LPY0	LIGHT PEN Y
21	15	SE7							SE0	SPRITE ENABLE (ON/OFF)
22	16	N.C.	N.C.	RST	MCM	CSEL	XSCL2	XSCL1	XSCL0	X SCROLL MODE
23	17	SEX7							SEX0	SPRITE EXPAND Y
24	18	VS13	VS12	VS11	VS10	CB13	CB12	CB11	N.C.	Memoria video
25	19	IRQ	N.C.	N.C.	N.C.	LPIRQ	ISSC	ISBC	RIRIQ	Interrupt Request's
26	1A	N.C.	N.C.	N.C.	N.C.	MLPI	MISSC	MISBC	MRIRQ	Interrupt Request MASKS
27	1B	BSP7							BSP0	Priorità Sfondo Sprite
28	1C	SCM7							SCM0	Selezione Sprite Multicolore
29	1D	SEXX7							SEXX0	Espansione X dello sprite
30	1E	SSC7							SSC0	COLLISIONE sprite-sprite
31	1F	SBC7							SBC0	COLLISIONE sprite-sfondo
INFORMAZIONI-COLORE										
32	20									Margine
33	21									Sfondo 0
34	22									Sfondo 1
35	23									Sfondo 2
36	24									Sfondo 3

37 25						Sprite			SMC 0
38 26						Multicolori			SMC 1
39 27									Col.Sprite 0
40 28									Col.Sprite 1
41 29									Col.Sprite 2
42 2A									Col.Sprite 3
43 2B									Col.Sprite 4
44 2C									Col.Sprite 5
45 2D									Col.Sprite 6
46 2E									Col.Sprite 7

Per il multicolore fare riferimento al codice 0...7.
Per informazioni del colore vedere la tabella a Pag. 139.

APPENDICE O

CONTROLLORE DEL SUONO DEL COMMODORE 64

Questa tabella contiene i valori di cui si ha bisogno per generare un programma a 3 voci, con il COMMODORE 64.
Si ha bisogno del comando:

POKE (registro), (numero)

Per il registro diviso in Hi-byte e Lo-byte bisogna addizionare i valori scelti, cioè per la VOCE 2 di ATTACK/DECAY

POKE 54272+12,5*16+7 OPPURE POKE 54284,87

Da notare che anche il volume deve essere predisposto per poter generare un tono.
POKE 54296 seguito da un numero compreso tra O e 15 controlla il volume per tutte le voci.

Registro di controllo del SID: 54272(Dec) - D4OO(Esa)

REGISTRI **COMMENTO**

1	2	3	
0	7	14	FREQUENZA, LO-BYTE (0...255)

1	8	15	FREQUENZA, HI-BYTE (0...255)				
2	9	16	TOCCO SIMULATO (volume), LO-BYTE (0...255) (Solo rettangolare)				
3	10	17	TOCCO SIMULATO (volume), HI-BYTE (O... 15)				
4	11	18	Forma d'onda:	RUMORE	REUANGOLARE	DENTE DI SEGA	TRIANGOLARE
				129	65	33	17
5	12	19	ATTACK				DECAY
			0*16 (duro)... 15*16 (debole)				0 (duro)... 15 (debole)
6	13	20	SUSTAIN				RELEASE
			0*16 (muto)... 15*16 (forte)				0 (veloce)... 15 (lento)
24	24	24	VOLUME: 0 (muto)... 15 (pieno)				

ESEMPIO: Nota triangolare continua della VOCE 2

S1=54272
POKE SI+24,15:POKE SI+7, 207:POKE SI+8,34:POKE SI+13,240
(volume) (frequenza Lo) (frequenz HI) (volume alto continuo)

Posizione ON del tono: **POKE SI+11,17**
Posizione 0FF del tono: **POKE 51+11,0**

I DIVERSI REGISTRI DEL SID

REGISTRO	**CONTENUTO**				
21	LIMITE FILTRO Dl FREQUENZA, LO-BYTE (0... 7)				
22	LIMITE FILTRO DI FREQUENZA, HI-BYTE (0... 255)				
23	RISONANZA	ACCENZIONE DEL FILTRO			
	0(minima)... 15*16 (massima)	esterno	Sti 3	Sti 2	Sti 1
		8	4	2	1

24	MODO-FILTRI				VOLUME
	Sti 3 aus	Hoch Pass	Band Pass	Tief Pass	
	128	64	32	16	0(debole)... 15(forte)

Addizionalmente il SID ha altri 4 registri che non sono usati nella generazione della nota.
La CPU può leggere alcune informazioni in questi punti.

REGISTRO	CONTENUTO
25	PADDLE X
26	PADDLE Y
27	OSCILLATORE 3
28	ASDR 3

Per leggere le paddle dalla porta di controllo battere:

SI 54272:X=PEEK(SI+25):Y=PEEK(SI+26)

x e y contengono i valori da 0 a 255 dipendente dallo stato interno dei paddle.
Il registro 27 e 28 contiene il valore momentaneo della VOCE 3 e dell'ADSR 3 in modo da creare numeri random (a caso) o influenzare le rimanenti voci per effetti speciali.

Usando queste istruzioni si possono imitare alcuni strumenti

Strumenti	**Forma d'onda**	**Att/Decay**	**Sust/Release**	**Duty-cicle**
Pianoforte	Pulse 65	9	0	Hi-0, Lo-255
Flauto	Triangolare 17	96	0	
Cembalo	Dente sega 33	9	0	
Xilofono	Triangolare 17	9	0	
Organo	Triangolare 17	0	240	
Fisarmonica	Triangolare 17	102	0	
Tromba	Dente sega 33	96	0	

Note: bisogna disporre il primo POKE prima del POKE della forma d'onda.

APPENDICE P

VALORE DELLE NOTE MUSICALI

In questa appendice sono contenuti una lista delle NOTE # ed i rispettivi valori da inserire nelle istruzioni POKE dei registri di HI-Byte e LO-Byte del SID, per produrre le note indicate.

Nr.	Note-Ottave	Frequenza (Hz)	Parametri	HI-Byte	LO-Byte
0	C-0	16.4	278	1	22
1	C#-0	17.3	295	1	39
2	D-0	18.4	313	1	57
3	D#-0	19.4	331	1	75
4	E-0	20.6	351	1	95
5	F-0	21.8	372	1	116
6	F#-0	23.1	394	1	138

7	G-0	24.5	417	1	161
8	G#-0	26.0	442	1	186
9	A-0	27.5	468	1	212
10	A#-0	29.1	496	1	240
11	H-0	30.9	526	2	14
12	C-1	32.7	557	2	45
13	C#-1	34.6	590	2	78
14	D-1	36.7	625	2	113
15	D#-1	38.9	662	2	150
16	E-1	41.2	702	2	190
17	F-1	43.7	743	2	231
18	F#-1	46.2	788	3	20
19	G-1	49.0	834	3	66
20	G#-1	51.9	884	3	116
21	A-1	55.0	937	3	169
22	A#-1	58.3	992	3	224
23	H-1	61.7	1051	4	27
24	C-2	65.4	1114	4	90
25	C#-2	69.3	1180	4	156
26	D-2	73.4	1250	4	226
27	D#-2	77.8	1325	5	45
28	E-2	82.4	1403	5	123
29	F-2	87.3	1487	5	207
30	F#-2	92.5	1575	6	39
31	G-2	98.0	1669	6	133
32	G#-2	103.8	1768	6	232
33	A-2	110.0	1873	7	81
34	A#-2	116.5	1985	7	193
35	H-2	123.5	2103	8	55
36	C-3	130.8	2228	8	180
37	C#-3	138.6	2360	9	56
38	D-3	146.8	2500	9	146
39	D#-3	155.6	2649	10	89
40	E-3	164.8	2807	10	247
41	F-3	174.6	2974	11	158
42	F#-3	185.0	3150	12	78
43	G-3	196.0	3338	13	10
44	G#-3	207.7	3536	13	208
45	A-3	220.0	3746	14	162
46	A#-3	233.1	3969	15	129
47	H-3	246.9	4205	16	109
48	C-4	261.6	4455	17	103
49	C#-4	277.2	4720	18	112
50	D-4	293.7	5001	19	137
51	D#-4	311.1	5298	20	178
52	E-4	329.6	5613	21	237
53	F-4	349.2	5947	23	59
54	F#-4	370.0	6301	24	157
55	G-4	392.0	6676	26	20
56	G#-4	415.3	7072	27	160
57	A-4	440.0	7493	29	69
58	A#-4	466.2	7939	31	3
59	H-4	493.9	8411	32	219
60	C-5	523.3	8911	34	207
61	C#-5	554.4	9441	36	225
62	D-5	587.3	10002	39	18
63	D#-5	622.3	10597	41	101
64	E-5	659.3	11227	43	219

65	F-5	698.5	11894	46	118
66	F#-5	740.0	12602	49	58
67	G-5	784.0	13351	52	39
68	G#-5	830.6	14145	55	65
69	A-5	880.0	14986	58	138
70	A#-5	932.3	15877	62	5
71	H-5	987.8	16821	65	181
72	C-6	1046.5	17821	69	157
73	C#-6	1108.7	18881	73	193
74	D-6	1174.7	20004	78	36
75	D#-6	1244.5	21193	82	201
76	E-6	1318.5	22454	87	182
77	F-6	1396.9	23789	92	237
78	F#-6	1480.0	24203	98	115
79	G-6	1568.0	26702	104	78
80	G#-6	1661.2	28290	110	130
81	A-6	1760.0	29972	117	20
82	A#-6	1864.7	31754	124	10
83	H-6	1975.5	33642	131	106
84	C-7	2093.0	35643	139	59
85	C#-7	2217.5	37762	147	130
86	D-7	2349.3	40008	156	72
87	D#-7	2489.0	42387	165	147
88	E-7	2637.0	44907	175	107
89	F-7	2793.8	47578	185	218
90	F#-7	2960.0	50407	196	231
91	G-7	3136.0	53404	208	156
92	G#-7	3322.4	56580	221	4
93	A-7	3520.0	59944	234	40
94	A#-7	3729.3	63508	248	20

I seguenti parametri sono indicativi.
Usando più voci sarebbe meglio di «stonare» la seconda e terza voce, per ottenere un suono migliore.

APPENDICE Q

Mappa di memoria del Commodore 64

(* = indirizzi utilizzabili)

Hex	Decimale	Commento
0000	0	Direzione data registro 6510
0001	1	Data buffer 6510
0002	2	Non usato
0003 - 0004	3 - 4	Vettore di conversione del floating point. Integer
0005 - 0006	5 - 6	Vettore di conversione dell'integer. Floating point
0007	7	Byte di ricerca
0008	8	Quote mode flag
0009	9	Contatore del TAB delle colonne
000A	10	0 = LOAD, 1 = VERIFY
000B	11	Puntatore di input del buffer/numero elementi
000C	12	Flag per standard DIM
000D	13	Tipo: FF = stringa, 00 = numerico
000E	14	Tipo: 80 = integer, 00 = floating point
000F	15	Flag per DATA / LIST
0010	16	Elementi / FN * Flag

	0011	17	00 = INPUT, 40 = GET, 98 = READ
	0012	18	Segno per ATN
	0013	19	Effettivo I/O-device
*	0014 - 0015	20 - 21	Valore dell'integer
	0016	22	Puntatore per l'effettiva stringa stack
	0017 - 0018	23 - 24	Ultimo vettore stringa temporaneo
	0019 - 0021	25 - 33	Stack per stringhe temporanee
	0022 - 0025	34 - 37	Pointer area ausiliaria
	0026 - 002A	38 - 42	Prodotto di moltiplicazione
*	002B - 002C	43 - 44	Puntatori Inizio del programma
*	002D - 002E	45 - 46	Puntatori Inizio delle variabili
*	002F - 0030	47 - 48	Puntatori Inizio degli arrays
*	0031 - 0032	49 - 50	Puntatori Fine degli arrays
*	0033 - 0034	51 - 52	Puntatori Inizio delle stringhe (verso il basso)
	0035 - 0036	53 - 54	Puntatori di striga ausiliario
*	0037 - 0038	55 - 56	Fine della memoria
	0039 003A	57 - 58	Numero delle effettive linee-basic
	003B - 003C	59 - 60	Numero delle effettive linee-basic
	003D 003E	61 - 62	Cont dei pointer
	003F - 0040	63 - 64	Numero delle effettive linee di data
	0041 0042	65 - 66	Effettivi indirizzi data
*	0043 - 0044	67 - 68	Jump vector per Input
	0045 - 0046	69 - 70	Nomi variabili effettivi
	0047 - 0048	71 - 72	Indirizzi variabili effettivi
	0049 - 004A	73 - 74	Puntatore FOR-NEXT
	004B - 004C	75 - 76	Puntatore basic per la scratch area
	004D	77	Accumulatore per la comparazione dei simboli
	004E - 0053	78 - 83	Scratch area
	0054 - 0056	84 - 86	Vettore di funzione jump
	0057 - 0060	87 - 96	Scratch per operazioni numeriche
*	0061	97	Accumulatore del floating point #1 (FAC): Esponente
*	0062 - 0065	98 - 101	Accumulatore del floating point #1 (FAC): Mantissa
*	0066	102	Accumulatore del floating point #1 (FAC): Segno
	0067	103	Polynom pointer
	0069 - 006E	105 - 110	FAC #2
	006F	111	Comparazione di segno FAC1/2
	0070	112	Least significant digit FAC41 (rounding)
	0071-0072	113 - 114	Lunghezza del cassette buffer
*	0073-008A	115 - 138	CHRGET subroutine (immette un carattere)
	007A - 007B	122 - 123	Puntatore per subroutine basic
	008B - 008F	139 - 143	Inizio volume del RND
*	0090	144	Status byte ST
	0091	145	Flag per il tasto STOP e RVS
	0092	146	Costante per cassette timeout
	0093	147	0 = LOAD; 1 = VERIFY
	0094	148	Output seriale: flag form waithing carattere
	0095	149	waithing carattere
	0096	150	EOT fornito alle cassette
	0097	151	Memoria di registro
*	0098	152	Numero di file aperti
*	0099	153	Input device (normale = 0)
*	009A	154	Output device (CMD)
	009B	155	Byte polarity dal nastro
	009C	156	Flag per il byte ricevuto
	009D	157	Controllo OUTPUT (80 = diretto,00 = RUN)
	009E	158	Errore sul buffer carattere dalla cassetta 1

	009F	159	Errore dalla cassetta corretto
*	00A0 - 00A2	160 - 162	Clock interno HMS
	00A3	163	Contatore seriale di bit Flag EOI
	00A4	164	Ciclo del contatore
	00A5	165	Contatore in basso quando scrive su cassette
	00A6	166	Puntatore per il buffer della cassetta
	00A7 - 00AB	167 - 171	Flags per essere scritte e lette da cassetta
	00AC - 00AD	172 - 173	Puntatore per l'inizio del programma
	00AE - 00AF	174 - 175	Puntatore per la fine del programma
	00B0 - 00B1	176 - 177	Costante di tempo (time-costant) del nastro
*	00B2 - 00B3	178 - 179	Puntatore d'inizio per il buffer della cassetta
	00B4	180	Temporizzatore del nastro (1 = disposto), contatore bit
	00B5	181	EOT del nastro/RS 232 manda il bit seguente
	00B6	182	***
*	00B7	183	Numero di caratteri nel nome del file
*	00B8	184	Effettivo Log. del nome del file
*	00B9	185	Effettivo indirizzo secondario
*	00BA	186	Effettivo device
*	00BB - 00BC	187 - 188	Puntatore del nome del file
	00BD	189	***
	00BE	190	Numero di blocchi relativi al Read/Write
	0OBF	191	Buffer di parole seriale
	00C0	192	Flag del motore della cassetta
	00C1 - 00C2	193 - 194	Inizio indirizzi I/O
	00C3 - 00C4	195 - 196	Puntatore del KERNEL, vettore degli indirizzi
*	00C6	198	Numero di caratteri nel buffer della tastiera
*	00C7	199	
	00C8	200	Linea e puntatore per input
	00C9 - 00CA	201 - 202	Posizione del cursore (Righe, colonne)
*	00CB	203	Tasto premuto (64=non è un tasto)
	00CC	204	
	00CD	205	Contatore per il lampeggio del cursore
	00CE	206	Carattere sotto la posizione di cursore
	00CF	207	Cursore nella fase di lampeggio
	00D0	208	lnput da tastiera/schermo
*	00D1 -00D2	209 - 210	Puntatore delle righe di schermo
*	00D3	211	Puntatore delle colonne di schermo
	00D4	212	0 = cursore diretto, altrimenti programmato (QUOTE MODE)
*	00D5	213	Lunghezza delle effettive linee di schermo (40/80)
*	0006	214	Linee dove il cursore risiede
	00D7	215	Ultimo tasto/checksum/buffer
*	00D8	216	Numero di inserimenti rimasti
*	00D9 - 00F1	217 - 240	Tavola delle linee di schermo
	00F0	241	Linee di schermo non vere
	00F2	242	Label delle linee di schermo
*	00F3 - 00F4	243 - 244	Puntatore del colore
	00F5 - 00F6	245 - 246	Puntatore della tabella della tastiera
	00F7 - 00F8	247 - 248	Puntatore Receive RS232
	00F9 - 00FA	249 - 250	Puntatore Transmit RS232
*	00FB - 00FE	251 - 254	Spazio rimasto in pagina = 0
	00FF	255	Basic RAM
	0100 - 010A	256 - 266	Scratch RAM per convertire il formato del floating Puntatore in ASCII
	0100 - 013E	256 - 318	Errore di nastro
	0100 - 01FF	256 - 511	CPU Stack

*	0200 - 0258	512 - 600	Input buffer basic
*	0259 - 0262	601 - 610	Tabella per file logici
*	0263 - 026C	611 - 620	Tabella per numeri di divice
*	0280 - 0276	621 - 630	Tabella per indirizzi secondari
*	0277 - 0280	631 - 640	Buffer di tastiera
*	0281 - 0282	641 - 642	Inizio RAM per sistema operativo
*	0283 - 0284	642 - 644	Fine RAM per sistema operativo
	0285	645	Time out flag per bus seriale
*	0286	646	Codice colore effettivo
	0287	647	Colore sotto cursore
*	0288	648	Pagina indirizzi video RAM
*	0289	649	Massima SRE del buffer di tastiera
*	028A	650	Ripetizione tastiera (128 = tutti i tasti)
*	028B	651	Contatore per l'incremento della ripetizione
	028C	652	Contatore per il ritardo della ripetizione
*	028D	653	Flag per lo SHIFT/CNTRL
	028E	654	Ultimo SHIFT pattern della tastiera
	028F - 0290	655 - 656	Puntatore decodificatore della tavola della tastiera
*	0291	657	SHIFT mode (0 = disposto, 128=bloccato)
	0292	658	scrolling automatico verso il basso (0 = ON: 0 = OFF)
	0293	659	Registro di controllo RS232
	0294	660	Registro di comando RS232
	0295 - 0296	661 - 662	Non standard (Bit time) ??
	0297	663	Status registro RS232
	0298	664	Numero di bit da trasmettere
	0299 - 029A	665 - 666	Baud Rate
	029B	667	RS232 Puntatore Receive receive
	029C	668	RS232 Puntatore Input
	029D	669	RS232 Puntatore Transmit
	029E	670	RS232 Puntatore Output transmit
	029F - 02A0	271 - 672	contiene il vettore IRQ durante l'operazione cassette
	02A1 - 02FF	673 - 767	***
*	0300 - 0301	768 - 769	Vettore di messaggio di errore
	0302 - 0303	770 - 771	Vettore di warm start
	0304 - 0305	772 - 773	Convertitore di parole in Token
	0306 - 0307	774 - 775	Convertitore di Token nella tastiera
	0308 - 0309	776 - 777	Esegue il prossimo comando BASIC
	030A - 030B	778 - 779	Alimenta elementi aritmetici
	030C	780	Memoria per il 6502,, Registro A
	030D	781	Memoria per il 6502,, Registro X
	030E	782	Memoria per il 6502,, Registro Y
	030F	783	Memoria per il 6502,, Registro P
	0310 - 0313	784 - 787	USR jump
	0314 - 0315	788 - 789	Hardware Interrupt (IRO) (EA31)
	0316 - 0317	790 - 791	Break Interrupt (FE66)
	0318 - 0319	792 - 793	Non maskable Interrupt (NMI) (FE47)
	031A - 031B	794 - 795	OPEN (F40A) (F34A)
	031C - 031D	796 - 797	CLOSE (F291)
	031E - 031F	798 - 799	Canale di Input (F2C7) (F209)
	0320 - 0321	800 - 801	Canale Output (F250)
	0322 - 0323	802 - 803	Registro I/O
			CLEAR tutti i canali aperti (F333)
	0324 - 0325	804 - 805	INPUT (F157)
	0326 - 0327	806 - 807	OUTPUT (F1CA)
	0328 - 0329	808 - 809	STOP-tasto controllo (F770) (F6ED)
	032A - 032B	810 811	GEI (F13E)
	032C - 032D	812 - 813	Chiude tutti i canali (F32F)

032E - 032F	814 - 815	USER IRQ (FF66)
0330 - 0331	816 - 817	RAM bloccate (F4A5)
0332 - 0333	818 - 819	RAM SAVE (F5ED)
0334 - 033B	820 - 827	***
033C - 03FB	828 - 1019	Buffer di cassetta
0400 - 07FF	1024 - 2047	1 K memoria video
(0400 - 07E7	1024 - 2023	(video matrix)
(07F8 -07FF	2040 - 2047	Puntatore di SPRITE
0800 - 9FFF	2048 - 40959	RAM usata per il BASIC
A000 - BFFF	40960 - 49151	8 Basic ROM
C000 - CFFF	49152 - 53247	4K RAM

INDICE ALFABETICO

La Commodore si augura che abbiate apprezzato questa GUIDA all'USO del COMMODORE 64. Quantunque questo manuale contenga qualche informazione e suggerimenti di programmazione NON È tuttavia inteso come manuale di riferimento per il programmatore. Per i avanzati e per gli appassionati la Commodore suggerisce di prendere in considerazione l'acquisto della GUIDA DI RIFERIMENTO PER IL PROGRAMMATORE COMMODORE 64 disponibile attraverso il rivenditore locale Commodore.

Oltre agli aggiornamenti e alle correzioni, nelle riviste COMMODORE e POWER PLAY sono disponibili suggerimenti e consigli per l'uso del database Commodore della COMPUSERVE INFORMATION NETWORK, cui si accede attraverso un VICMODEM.

TABELLA RAPIDA DI RIFERIMENTO COMMODORE 64

VARIABILI SEMPLICI

Tipo	*Nome*	*Campo*
Reali	XY	+/- 1.70141183E+38 +/- 2.93873588E-39
Interi	XY%	+/- 32767
Stringa	XY$	Da 0 a 255 caratteri

X è una lettera (A-Z). Y è una lettera o un numero (da 0 a 9). I nomi variabili possono avere più di due caratteri ma vengono riconosciuti soltanto i primi due.

VARIABILI MATRICE

Tipo	*Nome*
Dimensione singola	XY(5)
Due dimensioni	XY(5,5)
Tre dimensioni	XY(5,5,5)

Dove occorre possono essere usate matrici con un massimo di 11 elementi (indice da 0 a 10). Le matrici con più di 2 elementi devono essere dimensionate (DIM)

OPERATORI ALGEBRICI

=	Assegna un valore alla variabile
-	Negazione
^	Elevamento ad esponente
*	Moltiplicazione
/	Divisione

+ Somma
- Sottrazione

OPERATORI LOGICI E RELAZIONALI

=	Uguale
<>	Diverso da
<	Minore di
>	Maggiore di
<=	Minore di o uguale a
<=	Maggiore di o uguale a
NOT	«NOT» logico
AND	«AND» logico
OR	«OR» logico

L'espressione è uguale -1 se vera, a 0 se falsa

COMANDI DI SISTEMA

LOAD "NAME"	Carica un programma da nastro
SAVE "NAME"	Salva un programma su nastro
LOAD "NAME",8	Carica un programma da disco
SAVE "NAME",8	Salva un programma su disco
VERIFY "NAME"	Verifica che il programma è stato salvato (SAVE) senza errori
RUN	Esegue un programma
RUN xxx	Esegue un programma iniziando alla riga xxx
STOP	Interrompe l'esecuzione
END	Termina l'esecuzione
CONT	Continua l'esecuzione del programma dalla riga in cui il programma si è interrotto
PEEK(X)	Dà i contenuti della locazione di memoria di X
POKE X,Y	Cambia i contenuti della locazione X al valore Y
SYS xxxxx	Salta un programma in linguaggio macchina iniziando in xxxxxx
WAIT X,Y,Z	Il programma attende fino a che i contenuti della locazione X, dove è stato eseguito EOR con Z e sommato logicamente con Y (AND) sono diversi da 0
USR(X)	Trasmette il valore di X alla subroutine in linguaggio macchina

COMANDI DI EDITING E DI FORMATTAZIONE

LIST	Lista l'intero programma
LIST A-B	Lista dalla riga A alla riga B
REM Message	Un messaggio di commento può essere elencato me viene ignorato con l'esecuzione del programma
TAB(X)	Usato nelle istruzioni PRINT. Spazia X posizioni sullo schermo
SPC(X)	Stampa (PRINT) X spazi vuoti sulla riga
POS(X)	Da la posizione corrente dei cursore
CLR/HOME	Posiziona il cursore all'angolo superiore sinistro dello schermo

COMANDI DI EDITING E DI FORMATTAZIONE

SHIFT CLR/HOME	Cancella Io schermo ed inserisce il cursore in posizione di partenza
SHIFT INST/DEL	Inserisce spazi nella posizione corrente del cursore
INST/DEL	Cancella il carattere nella posizione corrente del cursore
CTRL	Quando usato con il tasto numerico dei colori, sceglie il colore di testo. Può essere usato nell'istruzione PRINT
Tasti CRSR	Sposta il cursore verso l'alto e verso il basso, verso sinistra o verso destra sullo

schermo

Tasto Commodore	Quando usato con SHIFT sceglie tra maiuscolo/minuscolo e modo schermo per grafici.
	Quando usato con un tasto di colore numerico, sceglie il colore di testo opzionale.

MATRICI E STRINGHE

DIM A(X, Y, Z)	Definisce gli indici massimi per A; riserva spazio per (X+1)*(Y+1)*(Z+1) elementi partendo da A(0,0,0)
LEN (X$)	Dà il numero dei caratteri in X$
STR$(X)	Dà il valore numerico di X, convertito in una stringa
VAL(X$)	Dà il valore numerico di A$ fino al primo carattere non numerico
CHR$(X)	Dà il carattere ASCII il cui codice è X
ASC(X$)	Dà il codice ASCII per il primo carattere di X$
LEFT$(A$,X)	Dà i caratteri X più a sinistra di A$
RIGHTS(A$,X)	Dà i caratteri X più a destra di A$
MID$(A$,X,Y)	Dà Y caratteri di A$ iniziando dal carattere X

COMANDI DI INPUT/OUTPUT

INPUT A$ o A	Stampa (PRINT) «?» sullo schermo ed attende che l'utente immetta una stringa o un valore
INPUT «ABC» ;A	Stampa (PRINT) un messaggio ed attende che l'utente immette un valore. Può anche essere INPUT A$
GET A$ o A	Attende che l'utente batta un valore ad un carattere; non occorre RETURN
DATA A; «B», C	Inizializza una serie di valori che possono essere usati dall'istruzione READ
READ A$ o A	Assegna il successivo valore DATA a A$ o A
RESTORE	Ripristina il puntatore di dati all'inizio leggendo (READ) in nuovo la lista DATA
PRINT «A= »; A	Stampa (PRINT) la stringa «A » e valore di A «;». Sopprime gli - «,» e tabula i dati al campo successivo

FLUSSO DEL PROGRAMMA

GOTO X	Salta alla riga X
IF A=3 THEN 10	IF (se) l'asserzione è vera THEN (allora) esegue la parte successiva dell'istruzione.
	IF (se) falsa esegue numero successivo di riga
FOR A=l TO 10 STEP 2: NEXT	Esegue tutte le istruzioni tra FOR ed il corrispondente NEXT, con A che ve da 1 a 10 a
	incrementi di 2. L'incremento è 1 salvo dove diversamente specificato
NEXT A	Definisce la fine dell'iterazione. A è facoltativo
GOSUB 2000	Salta alla subroutine iniziando alla riga 2000
RETURN	Contrassegna la fine delle subroutine. Ritorna all'istruzione che segue il GOSUB più recente
ON X GOTO A,B	Salta al numero di riga X-esimo sulla lista. Se X = 1 salta a A, ecc.
ON X GOSUB A,B	Salta alla subroutine alla riga X-esima nella lista

A PROPOSITO DELLA GUIDA PER L'USO DEL COMMODORE 64...

Colore eccezionale... sintesi sonora... grafici... capacità di calcolo... il matrimonio sinergetico della tecnologia più avanzata. Queste caratteristiche fanno del Commodore 64 il personal computer più avanzato nella sua classe.

La Guida d'uso Commodore 64 vi aiuta ad affrontare i problemi di calcolo anche se non avete mai usato prima d'ora un computer. Mediante istruzioni chiave dettagliate, potete dare uno sguardo al linguaggio BASIC ed al modo in cui Commodore 64 può essere messo al lavoro in una quantità di applicazioni.

Per coloro che hanno già familiarità con il microcomputer, le parti relative alla programmazione avanzata e le Appendici spiegano le caratteristiche migliorate del Commodore 64 ed il modo per ottenere il massimo delle sue capacità.

I Manuali di Ready 64 - http://ready64.org

Manuale d'Uso (versione 1.0)

CREDITI
Conversione realizzata da:
Eregil, Hiryu, Jacks, MarC=ello, Roberto, rust5.
Gruppo di lavoro formatosi nel forum di Ready64.

NOTE
Il gruppo di lavoro si è occupato della conversione del manuale in formato e-text e della correzione dei listati presenti nel manuale (segue lista dei cambiamenti).
Ci si riserva di aggiornare la presente versione del manuale, qualora si riscontrassero o ci venissero segnalati errori, inesattezze, imprecisioni.

NOTE alla conversione testuale
Correzioni apportate in BOUNCING BALL (listato di pag. 43):

- cambiato 40 in 39 nelle righe 30 e 80 per evitare che la pallina finisse nella prima colonna della seconda riga, non rimbalzando nel bordo di destra ma un carattere più in là
- cambiate le parentesi in virgolette nella REM di riga 40 per preservare il carattere grafico

Correzioni apportate in INDOVINELLO (listato di pag. 145):

- righe 250-340: alcune parole, una volta decodificate (ogni lettera codifica la lettera precedente: B sta per A, C sta per B ecc.), sono prive di senso in italiano nel listato originale

Correzioni apportate in SEQUENZA (listato di pag. 146):

- riga 830: NRXTI corretto in NEXTI
- aggiunta una riga 920 che controlli se si è risposto di sì a "vuoi giocare ancora?"

Correzioni apportate in PIANOFORTE ELETTRONICO (listato di pag. 147):

- righe 100-140: riallineato correttamente il disegno della tastiera del pianoforte
- riga 240: corretta la stampa degli spazi per cancellare il messaggio della riga 180
- riga 250: corretta l'assegnazione "Z=0"
- riga 280: corretto un pericoloso "POKES+23,7" in POKESI+23,7
- riga 530: la variabile OK era referenziata erroneamente come OC interferendo con il corretto funzionamento del tasto F7
- riga 580: corretto A$="" con A$=" " nella IF

Correzioni apportate in BNCPOKAL/BNCPOK2:

- cambiato 166 in 102 nelle righe 25 e 115: non ci vuole il codice CHR$, convertito nel codice POKE corrispondente.

Correzioni apportate alla Appendice O:

- Corretta la riga che descrive Attack/Decay